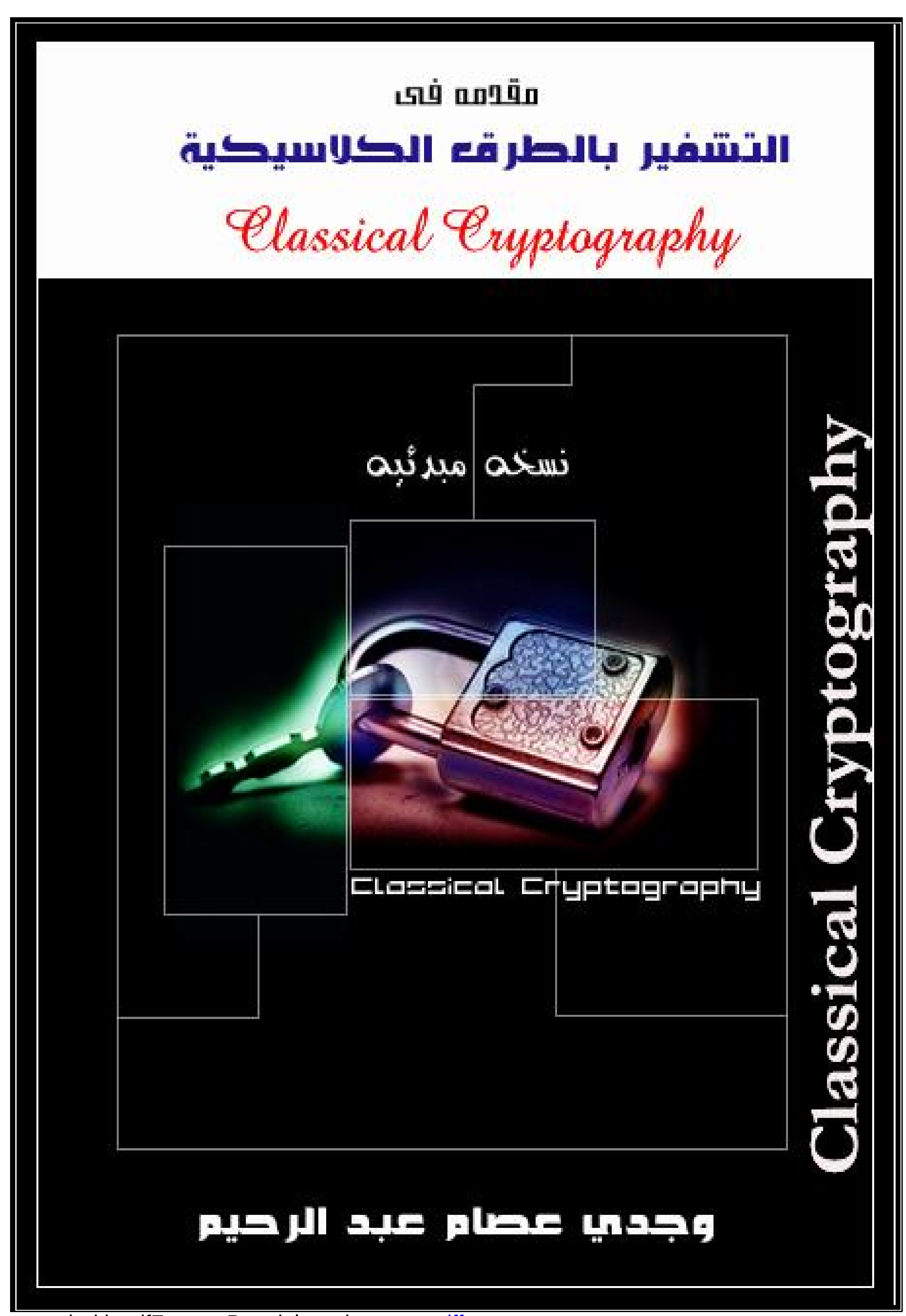


مقدمه فى
التشفير بالطرق الكلاسيكية
Classical Cryptography
نسخه مبدئيه
Classical Cryptography
Classical Cryptography
وجدي عصام عبد الرحيم

بسم الله الرحمن الرحيم

والصلاة والسلام على أشرف الخلق والمرسلين ، نبينا محمد عليه أفضل الصلاة والتسليم ، أما بعد ،

بين يديك أخي القارئ الكريم ، كتيب بسيط يتناول وبشكل سريع التشفير بالطرق الكلاسيكية ، والذي كان في الأصل عبارة عن مجموعه من الدروس قمت بوضعها في منتديات العاصفة www.3asfh.com ، وتم إعادة تنسيق محتويات الدروس ، مع اضافه العديد من النقاط الجديدة ، فشكر خاص لأعضاء ذلك المنتدى ، والقائمين عليه .

الطرق الكلاسيكية Classical Method هي عبارة عن طرق تشفير استخدمت منذ زمن بعيد وخاصة في أيام الحرب العالمية الأولى والثانية ، حيث كانت خطط الحرب وطرق الهجوم على العدو ترسل عن طريق رسائل عاديه مكتوبة بخط اليد (في الأغلب) ولكنها تشفر بأحد الطرق ، خوفا من أن تقع في أيدي العدو وبالتالي تفشل تلك الخطط .

بالرغم من أن تلك الطرق غير مجديه أبدا في الوقت الحالي ، فإنها موضوع هذا الكتيب. ربما تتساءل وما الجدوى من ذلك ، الجواب بكل بساطه ، لأنها تعتبر الأساس للكثير من الشفرات الحديثة ، اضافه إلى أن دراستها ينمى العقل على التفكير والبحث ، حيث أنها تعتمد على فقط على التلاعب بالأحرف (إما تبديل أماكن الأحرف **Transposition**، أو تبديلها بأحرف أخرى بعد عمليه حسابيه ما **Substitution**) ، غير ذلك فعملية كسر هذه الشفرات أمر في غاية المتعة .

ينقسم الكتيب إلى أربعة أقسام رئيسيه :

القسم الأول الأساسيات يتناول هذا القسم بعض من الخوارزميات والقواعد الرياضية الضرورية في عالم التشفير لبعض الطرق الكلاسيكية ، مثل نظريه القسمة ، الأعداد الأولية ، خوارزمية أقليدس ، خوارزمية اقليدس الممتدة .

القسم الثاني Introduction to Classical Cipher يتناول الشفرات الكلاسيكية وكيف يمكن التشفير وفك التشفير بتلك الطرق ، و يتناول وبشكل مبسط كيف يمكن كسر الشفرات **Cryptanalysis** لبعض من هذه الطرق .

القسم الثالث The Implementation يحتوي على أكواد برمجيه (مكتوبة بـ ++C) لتطبيق تلك الشفرات الموجودة بالفصل الثاني ، وكما يحتوي أكواد لبعض من الخوارزميات الرياضية الموجودة بالفصل الأول . (**البرامج كاملة موجودة مرفقه مع الكتيب**) .

القسم الرابع Introduction To Modern Cryptography ، يحتوي على مقدمه خفيفة "نظريه" للتشفير بالطرق الحديثة ، ويوضح الكثير من المصطلحات المستخدمة في عالم التشفير ، يعتبر قسم مهم لمن يريد البدء في عالم التشفير .

لعلك لاحظت أخي القارئ ، أن الكتيب حاليا نسخه مبدئية ، ويلزمه الكثير من الإضافات حتى يصبح مرجع ممتاز للطلبة والباحثين في هذا المجال ، فإذا كنت مهتم بهذا المجال ، ولديك القدرة على اضافه مواضيع ذات علاقة فأرجوا إعلامي وإرسال رسالة على بريدي ، وسنضمن لك كافه حقوقك الفكرية ، كما يرحب بكل نقد أو تعديل أو أي استفسار في هذا المجال .

أقسام سيتم إضافتها في النسخة القادمة :

Introduction To Number Theory and it's Application
Primality Testing , Factorization Techniques ,Congruence Solving, Random Number Generation

Introduction To Information & Coding Theory and it's Application
Huffman Code , Hamming Code , Error Correction Code , Entropy

Explain and Implementation for Most Crypto Cipher
RSA ,DES ,Triple DES, AES , Blowfish , ECC , ALGAMAL

History About Cryptography and Crypto Devices and Arabic Cryptographer (Alkindi, Taher Algamal)

Addition Classical Cipher & More About Cryptanalysis cipher

أخير ،أحب أن أشكر أخي "الصقر الجارح" من منتديات العاصفة ، على تصميمه للغلاف وإخراجه بهذا الشكل الرائع ، فجزاه الله خيرا . وأتمنى له دوام التوفيق في حياته .

للمراسلة أو السؤال أو الاستفسار : SudanGeek@hotmail.com
wajdyessam@hotmail.com

Romansy , at : 2-11-2007

القسم الأول : الأساسيات Preliminaries 7

خوارزمية القسمة THE DIVISION ALGORITHM 7

الأعداد الأولية Prime Number 8

القاسم المشترك الأعظم Greatest Common Divisor 10

خوارزمية أقليدس Euclidean Algorithm 11

الخوارزمية الإقليديه الممتدة Extended Euclidean Algorithm 12

النظرية الأساسية في الحساب The Fundamental Theorem of Arithmetic 16

المضاعف المشترك الأصغر Least Common Multiple 17

المعامل XOR – Exclusive-Or 17

Logarithms 18

القسم الثاني : الشفرات الكلاسيكية 20

الترميز Coding 21

التشفير بالطرق الكلاسيكية Classical Method 23

شفرات Monoalphabetic Substitution Cipher 24

شفره قيصر Caesar Cipher 24

شفره أتباش Atbash Cipher 29

شفره ROT13 29

شفره Affine Cipher 30

كسر الشفرات من نوع MONOALPHABETIC 32

شفرات Polyalphabetic substitution cipher 34

شفره Simple Shift Vigenere Cipher 34

كسر شفرات فجينير Vigenere البسيطة 37

طريقه كيسسكي KAISISKI لمعرفه عدد المفاتيح Key Length 38

طريقه فجينير الكاملة THE FULL VIGENERE CIPHER 42

شفره فجينير تلقائية المفتاح THE AUTO-KEY VIGENERE CIPHER 43

شفره فجينير طويلة المفتاح THE Running KEY VIGENERE CIPHER 44

شفرات PolyGram Substitution Cipher 45

شفره بلافير THE PLAYFAIR CIPHER 46

شفره هيل Hill Cipher 48

كسر الخوارزميات من نوع PolyGram Substitution Cipher 54

أسطوانة جيفيرسون THE JEFFERSON CYLINDER 54

التشفير بطريقه الـ HOMOPHONIC SUBSTITUTION CIPHERS 56

التشفير بالإبدال TRANSPOSITION CIPHERS 58
الشفره الأمنهTHE ONE-TIME PAD 61
أسئلة شامله لكل الموضوع 63

القسم الثالث : التطبيق Implementation 66

التعامل مع الحروف characters 67
إيجاد القاسم المشترك الأعظم Greatest Common Divisor 70
خوارزمية اقليدس الممتدة 71
اختبار أوليه العدد باستخدام Trial Division 72
التعامل مع اللوغريتم 72
Fast Exponentiation Algorithms 73
شفرة قيصر Caesar Cipher 74
شفرة ROT13 76
التشفير بطريقه Affine Cipher 77
شفرة فيجنير البسيطة Simple Shift Vigenere Cipher 78
شفره فيجينر الكاملة Full Vigenere Cipher 79
شفره فجينير تلقائية المفتاح Auto Key Vigenere Cipher 81
شفره فيجينر طويلة المفتاح the Running key Vigenere Cipher 82
شفرة بلافير Playfair Cipher 83
شفره Reverse Cipher 83

الفصل الرابع : مقدمه في التشفير بالطرق الحديثة 84

لماذا بالتشفير Why Cryptography 85
التشفير بالمفتاح المتناظر Symmetric key Cryptography 87
ما هو المفتاح ، وما هي أهميته 91
توليد المفتاح Key Generation 94
شفرات الكتلBlock Cipher 100
شفره التدفق Stream Ciphers 102
Block VS Stream من الأفضل 103
Triple DES 103
Advanced Encryption Standard 105
اداره المفتاح المتناظر Symmetric-Key Management 105
أجهزه حفظ المفاتيح Hardware-Based Key Storage 109
Crypto Accelerators 110

112 .... The Key Distribution Problem and Public-Key Cryptography
116 . لمحه تاريخية عن التشفير بالطريقة غير متناظرة History of Public-Key Cryptography
119 .... الخاتمة

القسم الأول : الأساسيات
PRELIMINARIES

القسم الأول : الأساسيات PRELIMINARIES

لفهم أغلب الشفرات الحديثة يلزم فهم الكثير من **نظريه الأعداد Number Theory** ، ولكن بما أن الكتيب يركز على الشفرات بالطرق الكلاسيكية فسوف نتناول ما يهمنا فقط في الوقت الحالي . وسوف تكون القواعد مكتوبة ولكن من غير إثبات رياضي لها Prove ، يمكنك تجاوز هذا القسم والعودة إليه لاحقا عندما تحتاجه في شفره Affine Cipher ، ولكني لا أفضل ذلك .

اذا كان لدينا عددين صحيحين a و b ، وكان a !=0 (لا يساوي 0) . نقول عن a يقسم b اذا كان لدينا عدد ثالث c بحيث b = a*c . اذا كان a يقسم b نشير إليه بالرمز a|b .

مثال بسيط :

3|27 صحيح ، لان 27 = 9*3 .

5|32 غير صحيحة ، لأن 32 = 4*c ولا توجد عدد صحيح يحل مكان c .

اذا كان لدينا ثلاثة أعداد صحيحة x,y,z ، وكان x|y و y|z اذا x|z .

مثال :

لدينا 3|9 ، و 9|72 ، اذا 3 تقسم 72 = 3|27

خوارزمية القسمة *THE DIVISION ALGORITHM*

وهي أحد الخوارزميات المهمة جدا ، حيث تقول انه يمكننا أن نمثل أي عدد صحيح ، وذلك بواسطة ضرب عدد صحيح b مع اضافه باقي r بحيث يكون الباقي عدد موجب وأقل من العدد b .

اذا كان لدينا عددين صحيح y,b ، وكان b أكبر من صفر ، اذا سيكون لدينا عددين q,r بحيث :

Y = b*q + r

q هو حاصل القسمة Quotient . r هو الباقي remainder . b هو المقسوم Divisor ، y هو القاسم dividend .

مثال لدينا المعادلة :

65 = 3*q + r

قيمه ال q هي 21 (وذلك بقسمه 65 على 3) ، والباقي r هو 2 .

3*21+2 وتساوي 65 .

مثال أخر :

$-21 = 5*q + r$

بقسمه 21- على 5 ، سوف نحصل على حاصل القسمة 4- ، والباقي سوف يكون 1- ، ولكن كما ذكرنا في القاعدة السابقة أن الباقي r دائما يكون موجب ، لذلك نقوم بجمع 5 في الباقي ، ونطرح 1 من حاصل القسمة

ليكون لدينا $q = -5$ ، $r = 4$.

نطبقها في المعادلة :

$-21 = 5*-5+4$ وهو صحيح .

الأعداد الأولية *Prime Number* :

تلعب الأعداد الأولية تلعب دورا كبيرا جدا في التشفير وخاصة في الطرق الحديثة ، وتعريفها كالتالي :

العدد الأولي ، هو العدد الصحيح integer أكبر من 1 ،ولا يقبل القسمة إلا على نفسه وعلى 1 . باقي الأعداد التي أكبر من 1 وغير أوليه تسمى أعداد مركبه Composite Number

مثال على أعداد أوليه :

2و 3 و 7 و 23 و29 و 163 والكثير غيرها .

مثال على أعداد مركبه :

4 (حيث أنها تقبل القسمة على 2)

100 (تقبل القسمة على 2 و 5) .

مثال على أعداد غير أوليه وغير مركبه

0 و 1 وجميع الأعداد السالبة ، مثل -21 .

جميع الأعداد الصحيحة التي أكبر من 1 ، لها قاسم أولي .
الأعداد الأولية غير منتهية .
اذا كان لدينا عدد صحيح مركب N ، اذا يكون لدى n قاسم أولي لا يتعدى الجذر التربيعي لN .

وهذا معناه اذا أردنا أن نعرف على العدد x هو أولي أم لا ، سوف نبحث من البداية 2 (لان واحد ليس أولي) إلى أن نصل إلى جذر العدد x . ونختبر كل عدد من هذه الأعداد ، هل يقبل x القسمة عليها ، في حال تحقق ذلك ، نكون قد عرفنا أن العدد ليس أولي ، وإذا لم يتحقق فيكون العدد أولي .

مثلا لدينا العدد 101 ، نبدأ بالاختبار من 2 ، إلى جذر 101 وهو 10 .
هل 101 يقبل القسمة على 2 . لا
هل 101 يقبل القسمة على 3 ، لا
هل 101 يقبل القسمة على 4 و5 و 6و 7 و8 و 9 . إلى أن نصل إلى 10 ، وأيضا لا يقبل ، اذا النتيجة أن العدد 101 هو عدد أولي .

هذه الطريقه في البحث ليست من أفضل الطرق في اختبار أوليه العدد ، وهناك الكثير من الطرق أفضل منها ، وهي تسمى طريقه **Trial Division** .

مثلا لدينا عدد ضخم يتكون من 500 خانه ، بعد أخذ الجذر التربيعي أصبح يتكون من 250 خانه ، الآن طريقه Trial Division سوف تكون مضيعه للوقت والجهد لأنها سوف تختبر من البداية وحتى ذلك العدد الذي يتكون من 250 خانه ، لذلك للتعامل مع الأعداد الضخمة (كما هو الحال في الشفرات الحديثة) يجب البحث عن حل أكثر كفائه .

وهناك الكثير من الطرق لهذا الأمر ، وسوف نتناولها في النسخة النهائية من الكتيب بإذن الله بالتفصيل ، ولكن يمكن لتسريع الأمر اختبار الأعداد الفردية فقط في طريقه Trial Division .

أيضا هناك طريقه **Sieve of Eratosthenes** ، وهي تعتمد على عمل إلغاء جميع مضاعفات الأعداد 2 و 3 و5 و7 من مدى الأعداد المراد البحث .

مثال للتوضيح ، نريد معرفه الأعداد الأولية بين 2 إلى 99 ، نقوم بعمل جدول فيه جميع تلك الأعداد (في الغالب يكون في مصفوفة) ، بعدها نقوم بشطب مضاعفات العدد 2 من الجدول ، ومضاعفات العدد 3 من الجدول وهكذا ، حتى يتبقى لدينا الجدول التالي :

		2	3		5		7		
	11		13				17		19
			23						29
	31						37		
	41		43				47		
			53						59
	61						67		
	71		73						79
			83						89
							97		

وهو الآن يحتوي على جميع الأعداد الأولية من 2 إلى 99 ، على العموم وكما لاحظت أنها سوف تستغرق مساحه كبيرة في حاله العدد المراد اختباره كبير ، ذلك هي غير مستخدمه بكثرة .

القاسم المشترك الأعظم *Greatest Common Divisor* (اختصارا *GCD*)

القاسم المشترك الأعظم لعددين هو أكبر عدد صحيح يقبل القسمة على العددين .

مثلا نريد معرفه القاسم المشترك الأكبر للعددين 30 ، 18 . نقوم بمعرفه جميع قواسم العددين ، ونأخذ العدد الأكبر من هذه القواسم :

30 يقبل القسمة على 1 و 2 و3 و 5 و 6 و 10 و 15 و 30
18 يقبل القسمة على 1 و 2 و3 و 6 و 9 و 18

نلاحظ في الأعداد السابقة أكبر قاسم يقبل القسمة على العددين وهو 6 .
GCD(30,18) = 6

يقال عن عددين "أوليان فيما بينهما" Relatively Prime اذا كان القاسم المشترك الأعظم لهم هو 1 .

الأزواج التالية من الأعداد أولينا فيما بينهما Relatively Prime :
8 و 9 ، لأن القاسم المشترك الأعظم لهم هو 1 .
23 و 44
27 و 55

الاشاره السالبة لا تؤثر في حساب القاسم المشترك الأعظم :
GCD(x,y) = GCD(x,-y) = GCD(-x,y) = GCD(-x,-y) = GCD(|x|,|y|)

مثال :
GCD(18,-54) = GCD(18,54) = 9

لأخذ القاسم المشترك الأعظم لمجموعه من الأعداد نقوم بأخذ القاسم المشترك الأعظم لعددين منهم ، والناتج نأخذه مع العدد الثالث وهكذا .

مثال : القاسم المشترك الأعظم ل :20 و 30 و 15 هو 5 وذلك :
GCD(20,30) = 10
CGD(10,15) = 5

اذا كان القاسم المشترك الأعظم لمجموعه من الأعداد = 1 ، وكان هناك زوج من هذه الأعداد (أي عددين) القاسم المشترك الأعظم هو غير 1 (أي ليس أوليان فيما بينهما) ، فأنها تسمى mutually relatively prime ، أما في حالة كان جميع الأزواج مع بعضها يكون القاسم يساوي واحد فإنها تسمى pairwise relatively prime .

مثال : لحساب القاسم المشترك الأعظم للأعداد 28 و 126 و 21 و 10 :

((28,126) , 21 , 10) =
(14 , 21 , 10) =
((14,21) , 10) =
(7,10) =
1 =

لاحظ النتيجة هي واحد ، بالرغم من أن هناك زوج من الأعداد غير أوليان فيما بينهما ، (28,126) = 7 .
وفي هذه الحالة تسمى مجموعه الأعداد mutually relatively prime . أما في حاله كان جميع الأزواج من الأعداد أوليان فيما بينهما فتسمى مجموعه الأعداد بـ pairwise relatively prime .

مثال :القاسم المشترك للأعداد 18 و 9 و 25 هو 1 ، ومع ذلك فهي mutually relatively prime ، لان القاسم المشترك الأعظم ل 18,9 هو 9 (أي هما ليسا أوليان فيما بينهما) .

خوارزمية أقليدس *Euclidean Algorithm*

اذا كان لدينا عددين c,q بحيث c = q*d + r ، اذا GCD(c,q) = GCD(d,r) .

القاعدة السابقة مهمة جدا ، ونستطيع من خلالها إيجاد القاسم المشترك الأعظم للعددين بسرعة .

مثال : أوجد القاسم المشترك الأعظم 132 و 55 باستخدام خوارزمية أقليدس :

132 = 55 * 2 + 22
55 = 22 * 2 + 11
22 = 11 * 2 + 0

نتوقف عند الوصول إلى الصفر ، ويكون القاسم المشترك الأعظم هو 11 وذلك :
GCD(132,55) = GCD(55,22) = GCD(22,11) = GCD(11,0) = 11

مثال أخر : أوجد GCD(252,198) باستخدام خوارزمية أقليدس ؟

252 = 198 * 1 + 54
198 = 54 * 3 + 36
54 = 36 * 1 + 18
36 = 18 * 2 + 0

نتوقف عند الصفر ، والقاسم هو 18 :

GCD(252,198) = (198,54) = (54,36) = (36,18) = (18,0) = 18

الخوارزمية الإقليديه الممتدة *Extended Euclidean Algorithm*

يمكن تمثيل القاسم المشترك الأعظم للعددين عن طريق دمج خطي Linear Combination مع عددين آخرين ، وذلك كالتالي : GCD(x,y) = m*x + n*y

كيف يمكن أيجاد قيمتي m و n وذلك عن طريق خوارزمية اقليدس الممتدة . وهناك ثلاثة طرق لمعرفة هذه القيم (الطرق هي مشابه لبعض ، لكن يمكن القول أنها مختصره من الأخريات) .

الطريقه الأولى ، وهي يمكن أن نطلق عليها التراجع Backward ، وهنا في هذه الطريقه نقوم بالحل عن طريق خوارزمية اقليدس وبعدها نقوم بالتراجع الخلفي ، لإيجاد قيم m و n . كما في المثال التالي :

مثال ، قم بتمثيل GCD(26,21) كLinear Combination للعددين 26 و 21 :

نبدأ في الحل كما هو الحال في طريقه اقليدس :

26 = 1* 21 + 5

21 = 4 * 5 + 1

5 = 5 * 1 + 0

ونتوقف عند الصفر . الآن المعادلة التي قبل المعادلة التي باقيها صفر (وهي في حالتنا هذه المعادلة الثانية) نقوم بكتابتها بهذا الشكل :

1 = 21 – 4 * 5 .... [1]

أيضا المعادلة الأولى بنفس الشكل :

5 = 26 – 1 * 21 ....[2]

الآن نعوض المعادلة [2] في [1] :

1 = 21 – 4 * (26 – 1 * 21)

ومن غير أجراء عمليه حسابيه ، فقط نفك القوس لينتج :

1 = 21 -4*26 +4*21

نجمع 21 + 4*21 ليكون لدينا الناتج النهائي :

1 = 5*21 + (-4)*26

نتأكد من النتيجة ، 26*4- + 21*5 والناتج يساوي واحد ، اذا المعادلة صحيحة .

اذا قيمة m هي 5 ، وقيمه n هي 4- .

(في الفصل القادم ، سنرى أن n و m يسمى معكوس العدد).

مثال : أوجد معكوس المعادلة a=3 MOD 26 ؟
الحل ، أولا نقوم بتطبيق خوارزمية اقليدس الممتدة ، على العددين 3 و 26 . كالتالي :

26=8*3+2 → 2= 26 - 8*3
3=1*2+1 → 1= 3 – 2
2=2*1+0

1= 3 - 2 = 3 - (26 - 8*3) = 3 – 26 + 8 * 3 = 9*3 – 26
1 = 9*3+(-1)*26

الآن العدد المجاور ل 3 هو المعكوس ، وهو 9 .

الطريقه الثانية ، وهي أسهل وأسرع بكثير ، وسوف نشرحها بنفس المثال السابق:
مثال ، قم بتمثيل GCD(26,21) كـLinear Combination للعددين 26 و 21 :
نقوم في البداية بإنشاء جدول ، ونضع هذه القيم فيه :

A	Q	X
26		
21		

الآن نبدأ في أخذ باقي قسمه 26 على 21 والناتج نضعه في نفس العمود أسفل 21 .
26 MOD 21 = 5
ونضع 5 أسفل 21 .

مره أخرى نأخذ باقي قسمه 21 و 5 والناتج نضعه أسفل 5 ، وهو 1 ، والمرة الأخيرة الباقي هو 0 وسوف نتوقف عنده .

A	Q	X
26		
21		
5		
1		
0		

الآن نقسم 26 على 21 ونضع الناتج في العمود q (بدأ من الصف الثاني) ، والناتج هو عدد كسري ، لكن نحن سوف نأخذ الجزء الصحيح وهو 1 ، أيضا نقسم 21 على 5 والناتج الصحيح هو 4 ، ونستمر هكذا .

A	Q	X
26		
21	1	
5	4	
1	5	
0		

الآن في العمود x ، نقوم بوضع أخر قيمتين 0 و 1 ، كما في الشكل ،

26		
21	1	
5	4	1
1	5	0
0		

الآن لكي نحسب الصف الثاني في العمود x ، نقوم بالأتي ، 4 = 0 + 1*4
أي سنضرب القيمتين اللتان يقعان في الصف الأسفل مباشره ونجمع الناتج مع القيمة في العمود x التي تأتي بعد صفين .

$4 = 4*1 + 0$

26		
21	1	4
5	4	1
1	5	0
0		

نفس الأمر مع السطر الأول : $5 = 4*1 + 1$

26		5
21	1	4
5	4	1
1	5	0
0		

الآن الخطوة الأخيرة :

A	Q	X
26		5
21	1	4

$1 = 5 * 21 - 4 * 26$
$1 = 5*21 + (-4)*26$

وبهذا نكون عرفنا معكوس العدد الأول والثاني (أو m و n) .

مثال / أوجد معكوس 23 MOD 26 ؟

A	Q	X
26		9
23	1	8
3	7	1
2	1	1
1		0

1 = 8 * 26 – 9 * 23. Correct

الاجابه صحيحة ، لكن ما هو معكوس 23 ؟

الجواب هو -9 ، وليس 9 .

1 = 8 * 26 + (- 9) * 23

الطريقه الثالثة ،وهي الأسهل برمجيا ، وطريقه هذه الخوارزمية كالتالي :

GCD (x, y) = snx + tny

يكون حساب قيمه s و t كما يلي :

$$s_j = s_{j-2} - q_{j-1}s_{j-1} \text{ for } j = 2, \ldots, n$$
$$s_0 = 1$$
$$s_1 = 0$$

$$t_j = t_{j-2} - q_{j-1}t_{j-1} \text{ for } j = 2, \ldots, n$$
$$t_0 = 0$$
$$t_1 = 1$$

مثال ، قم بتمثيل GCD(252,198) كـ Linear Combination للعددين 252 و 198 .

الآن كما هو موضح بالصورة أدناه ، العمود j يمثل عدد المراحل ، q يمثل حاصل القسمة ، r يمثل باقي القسمة وs,t هما المطلوبين .

j	q_j	r_j	s_j	t_j
0		252	1	0
1	1	198	0	1
2	3	54	$1-0\cdot1=1$	$0-1\cdot1=-1$
3	1	36	$0-1\cdot3=-3$	$1-(-1)\cdot3=4$
4	2	18	$1-(-3)\cdot1=4$	$-1-4\cdot1=-5$
5		0		

نقوم أولا بوضع العددين 252 و 198 في العموم r .

نقوم بقسمه 252 على 198 ونضع حاصل القسمة (عدد صحيح) في q وهو 1 ، نقوم بأخذ باقي قسمه 252 و 198 وتساوي 54 ونضعها في العمود r أسفل منهم .

نقسم 198 على 54 ونضع حاصل القسمة في q وهي 3 ، ونقوم بأخذ باقي قسمه 198 و 54 وتساوي 36 ونضعها في r .

الآن نقسم 54 على 36 ، حاصل القسمة 1 يكون في q ، باقي القسمة 18 يكون في r ، و نقسم 36 على 18 ، حاصل القسمة 2 يكون في q ، باقي القسمة 0 يكون في r .

وبما أن باقي القسمة يساوي صفر اذا نتوقف هنا . وننتقل إلى آخر عمودين وهما s ,t . وكما في القانون أعلاه دائما أول سطرين ثابتين : s0 = 1 , s1= 0 و t0 =0 , t1= 1 . نأتي لباقي السطور .

الآن لحساب السطر الثالث si ، نقوم بأخذ القيمة في السطر s i-2 (السطر الذي يسبقه بمرحلتين ، ثم نطرحه من حاصل ضرب السطر si-1 * qi-1 .

الآن السطر الثالث يساوي :

$$Si = si\text{-}2 - si\text{-}1*qi\text{-}1$$
$$= 1 - 0*1$$
$$= 1$$

وهكذا بالنسبة لباقي السطور ، ونفس الكلام بالنسبة للعامود t . إلى أن نصل إلى أخر قيمه لـs ولـt وهي : s=4 و t=-5 .

ويصبح حل السؤال هو :

$$GCD(252,198) = 4*252 + (-5)*198$$

قم بالتأكد ، عن طريق أجراء العملية الحسابية 4*252 + (-5)*198 وتساوي 18 وهو نفسه القاسم المشترك الأكبر للعددين 252 و 198 .

(الفائدة من هذه الطريقه أننا سنحصل على العددين s,t (وهو ما يسمى بالمعكوس) ، وهما مهمين جدا في طريقه التشفير بشفرة Affine Cipher ، والتي ستكون في القسم القادم).

النظرية الأساسية في الحساب *The Fundamental Theorem of Arithmetic*

تنص على أن أي عدد أكبر من 1 ، يمكن أن يكتب في الصوره : n = p1*p2*p3*…pn حيث P هو عدد أولي ، وهذا ما يعرف بتحليل العدد إلى عوامله الأولية Prime Power Factorization of an integer .

مثال :

$$24 = 2^3 \cdot 3$$
$$588 = 2^2 \cdot 3 \cdot 7^2$$
$$450 = 2 \cdot 3^2 \cdot 5^2$$

المضاعف المشترك الأصغر *Least Common Multiple* (اختصارا *LCM*)

وهو أقل عدد صحيح يقبل القسمة على العددين .
وأسهل طريقه لإيجاده عن طريق ضرب العددين وقسمة الناتج على القاسم المشترك الأعظم للعددين :
LCM(x,y) = x*y / GCD(x,y)

مثال :

- **lcm(36, 78)** = 36 · 78/(36, 78) = 36 · 78/6 = 6 · 78 = 468
- **lcm(21, 56)** = 21 · 56/(21, 56) = 21 · 56/7 = 3 · 56 = 168
- **lcm(100, 2050)** = 100 · 2050/(100, 2050) = 100 · 2050/50 = 2 · 2050 = 4100

المعامل *XOR – Exclusive-Or*

تستخدم هذه العملية الرياضية في اغلب طرق التشفير ، وطريقه عملها كالتالي :

Exclusive-Or		
a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

ويمكن اعتبارها على أنها عمليه جمع مع أخذ باقي القسمة على 2 .

0 + 0 MOD 2 = 0
0 + 1 MOD 2 = 1
1 + 0 MOD 2 = 1
1 + 1 MOD 2 = 0

بعض القوانين :

$a \oplus a = 0$
$a \oplus 0 = a$
$a \oplus 1 = \sim a$, where $\sim$ is bit complement.
$a \oplus b = b \oplus a$ (commutativity)
$a \oplus (b \oplus c) = (a \oplus b) \oplus c$ (associativity)
$a \oplus a \oplus a = a$
if $a \oplus b = c$, then $c \oplus b = a$ and $c \oplus a = b$.

طبعا الدالة XOR مفيدة جدا في الكثير من الحالات ، أغلب المبرمجين المبتدئين يعرفوا أنه يمكن عمل swap بين متغيرين بدون استخدام متغير ثالث :

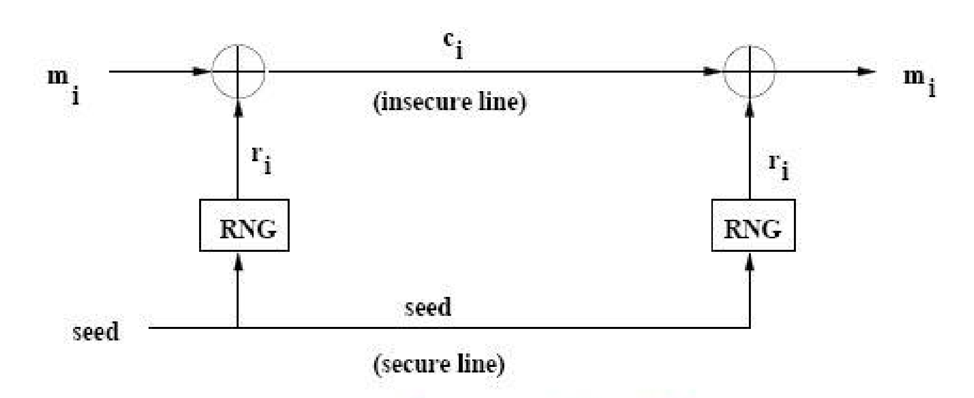
يعني بدل لما تكون العملية :

```
temp = a;
a = b;
b = temp;
```

تصبح :

```
a = a xor b;
b = a xor b;
a = a xor b;
```

اضافه إلى استخدامها في التشفير ، حيث تكون في الأغلب كما يلي :



Cryptosystem Using XOR.

C = r XOR p (r هو العدد العشوائي ، p هو الحرف الأصلي) .
ولفك التشفير :
P = r XOR c (c هو الحرف المشفر) .

على العموم، الدالة XOR لا تنفع للتشفير بذاتها ، بل يجب أن تستخدم ضمن طريقه ما .

An XOR might keep your kid sister from reading your files, but it won't stop a cryptanalyst for more than a few minutes

Logarithms

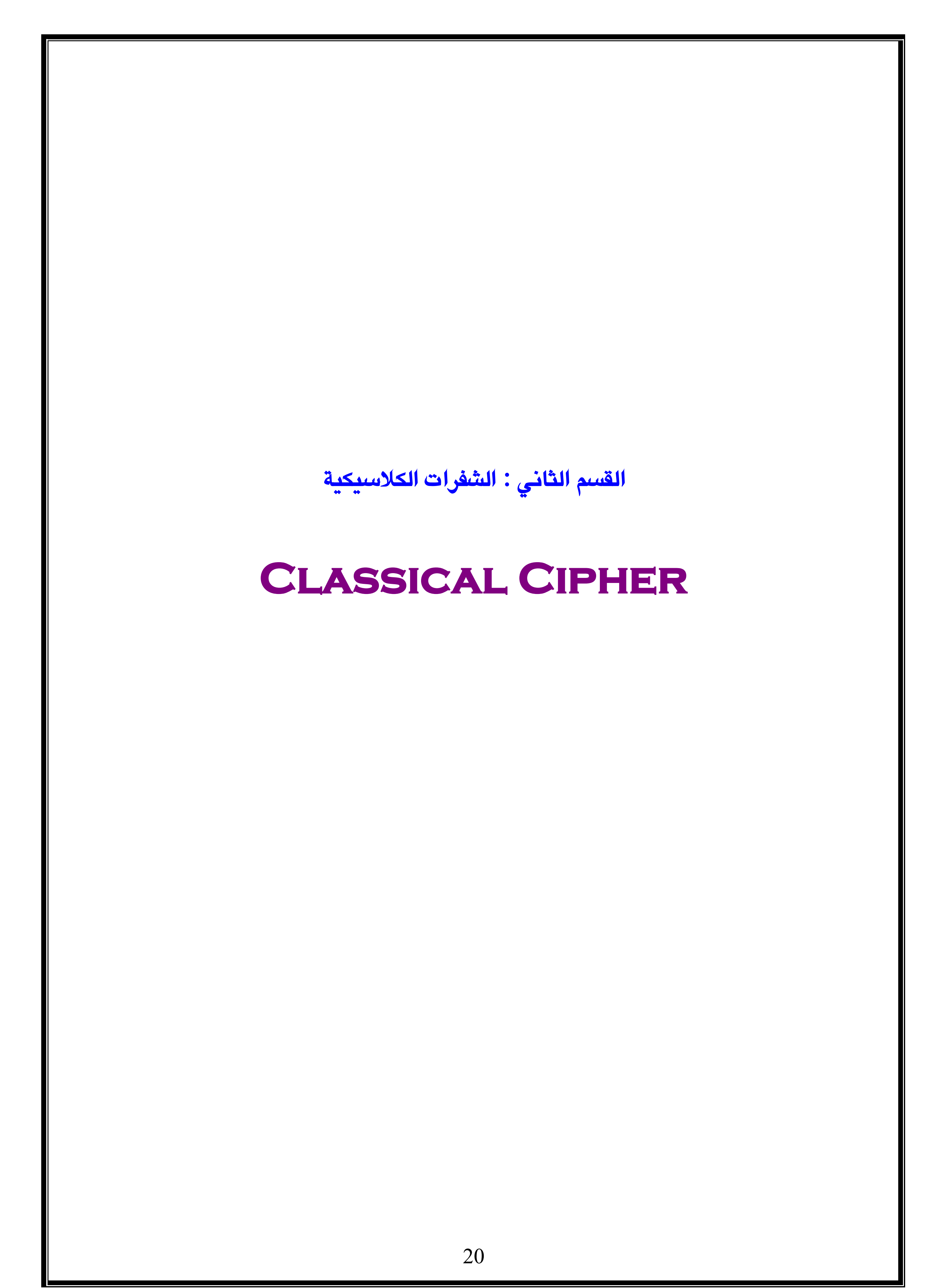
من المعروف أن $y = \log_b x$ وتساوي $b^y = x.$.

لكن المشكلة أنه أغلب الآلات الحاسبة وحتى لغة السي والجافا ، تحسب log على اساس log e اللوغريتم الطبيعي ، ولا يوجد بها log2 أو log10 .

فلتحويل من log e إلى log 2 ، نقوم : log2 x = log(x) / log(2)
ولتحويل من log e إلى log 10 ، نقوم : log10 x = log(x) / log(10)
ولتحويل من log10 إلى log2 ، نقوم : log2 x = 3.322 * log10 (x)

الفائدة من log2 هو أنه يخبرك كم bit يلزمك لتمثيل العدد x .
Log2(10000) = 13.28771
و هنا يخبرك بأنه يلزمك 14 بت لتمثيل العدد 10000

أيضا log10 يخبرك كم عدد يلزمك لتمثيل البتات x .



القسم الثاني : الشفرات الكلاسيكية

CLASSICAL CIPHER

الترميز *Coding* :

الترميز من المواضيع المهمة في عالم التشفير ، وذلك نظرا لسريه الشفرات التي تنتجها هذه العملية ، وبالرغم من ذلك فهي لم تستخدم بشكل كبير كما هو الحال مع التشفير **Encryption**، وذلك لما تتطلبه من إنتاج لغة سريه ، والاحتفاظ بها عند الأشخاص أن تتم عمليه الإرسال بينهم ، ومن أشهر هذا النوع **كتاب الرموز Codebook** .

في كتاب الرموز ، يكون لدينا جميع الكلمات المتوقع استخدامها في الإرسال ويقابل كل كلمة (تمثل النص الأصلي) كلمة أو عده كلمات أخرى تمثل الشفرة الناتجة من هذه الكلمة .

وعند تشفير كلمه ما بهذا الطريقة كل ما علينا هو البحث في كتاب الرموز واستخراج الكلمة المقابلة للكلمة المراد تشفيرها ، وهكذا حصلنا على الكلمة الجديدة المشفرة بهذه الطريقة

يوضح الشكل المقابل ، شكل مبسط لكتاب الرموز Code Book :

Codeword	Word
...	...
Computer	Dawn
...	...
Explode	Enemy
...	...
Lion	At
...	...
Run	Attack
...	...

مره أخرى نذكر أنه لتشفير كلمه معينه يجب أن تكون تحت العامود **Word** وبعدها ننظر إلى ما يقابلها في **Codeword** وأخذ الكلمة المشفرة.

لفك تشفير كلمة ما في Codeword ، كل ما علينا هو النظر إلى ما يقابلها في Word ، وسوف نحصل على الكلمة المطلوبة.

مثال ، لتشفير الجملة ألتاليه ، باستخدام كتاب الرموز السابق
ATTACK ENEMY AT DAWN

نقوم أولا بالنظر إلى كل كلمه وما يقابلها في كتاب الرموز Codebook ، لنحصل على الشفرة:
ATTACK يقابلها RUN
ENEMY يقابلها EXPLODE
ونستمر هكذا ، إلى أن نصل إلى الشفرة التالية :
RUN EXPLODE LION COMPUTER

وفي أغلب الأحيان في كتاب الرموز لن تجد الـ Codeword (أي أنك لن تجد كلمه ويقابلها كلمه مشفرة) لأنها من الممكن فك تشفيرها ، ويستخدم بدلا منها أرقام Code Number .

نأخذ المثال التالي ، ليبن كيف يمكن كسر الـ Codeword، وهذا المثال استخدم طريقه الـ **Jargon codes** وهي من أحد الطرق التي استخدمت بكثرة في الحرب العالمية الثانية ، حيث كان كانت ترسل الأوامر عبر الراديو (أو مذياع) الذي يكون مسموع للجميع حتى للأعداء.

لنفرض أن الطرف A قام بكتابه الرسالة ، وقام بتشفيرها باستخدام Codeword وقام بإرسالها إلى الطرف B عبر شبكه غير أمنه (أو أي وسيط إرسال أخر يحتمل أن يقرأه العدو) .
وكانت محتوى الرسالة بعد تشفيرها:
BOXER SEVEN SEEK TIGER5 AT RED CORAL

هذه الرسالة وبالرغم من أنها ليست واضحة قد يستفاد منها بطريقه ما ،

مثلا BOXER SEVEN قد يكون اسم لشخص ما تابع للطرف A ، الكلمة SEEK تدل مثلا على القبض الكلمة TIGER تدل على اسم العدو B ، الكلمة الأخيرة RED CORAL قد تدل على عنوان ما نظرا لأنها يسبقها AT .

اذا باستخدام الـ Codeword هناك احتمال كبير من التقاط الرسالة وفهم ولو أجزاء منها ، لذلك تم اللجوء إلى الحل الأخر وهو أن يستخدم الأرقام **CodeNumber** .

الآن وبعد إرسال الرسالة بترميزها بـ CodeNumber قد تصبح مثلا
85772 24799 10090 59980 12487

الرقم 85772 معناه BOXER SEVEN
والرقم 24799 معناه SEEK
وهكذا كل رقم يدل على كلمه ما أو عده كلمات وبالتالي سوف يصعب عمل كاسر الشفرة بكل تأكيد ، إن لم يكن يستحيل كسرها.

الآن لنعرف كتاب الرموز بشكل مبسط نقول هو جدول يحتوي على عامود Word وعامود أخر يمثل الـ CodeNumber أو CodeWord ، وفي حال التشفير ننظر إلى ما يقابل الكلمة في العامود الأخر ، ولفك التشفير نقوم بالعملية العكسية ، وهو ما يعرف بالـ **one-part-Code "كتاب رموز واحد"** .

ولأن كتاب الرموز حجمه كبير للغاية (لأنه يحتوي على جميع الكلمات التي يتوقع استخدامها في عمليه الإرسال ، وبعض الأحيان جميع الكلمات المعروفة تكون فيه) ، يكون مرتب بالترتيب الهجائي ، (مثلا) الحرف A يقابله في CodeNumber مثلا الرقم 20 ، الحرف B يقابله 21 ، الحرف C يقابله 22 ، سوف نلاحظ أنه يمكن لكاسر الشفرة كسر هذه الرموز لأنه يعرف أن الحرف A دائما أقل من الحرف Z وبقليل من المحاولة يتم كسرها ، لذلك تم اللجوء **إلى Two-Part-Code** وهنا يوجد كتاب للتشفير ، وكتاب أخر مرتب بطريقه أخرى لفك التشفير ، و هكذا لن يمكن تخمين ما ينتجه الكتاب الأول .

و على الرغم من سريه هذه الطريقة ، إلا إنها غير مجديه تماما ، نظرا لصعوبة الاحتفاظ بهذا الكتاب عند الطرفين ، وفي حال استخدمنا Two-Part-Code سنحتاج إلى كتابين عند كل طرف حتى يستطيعوا إرسال واستقبال الرسائل فيما بينهم ، ناهيك عن صعوبة إرسال الكتاب إلى الطرف الأخر والتأكد من أنه العدو لا يملك نسخه منه ، بالاضافه إلى صيانته (أضافه كلمات أخرى) وإرسالها مره أخرى للطرف الأخر.

التشفير بالطرق الكلاسيكية *Classical Method* :

الطرق الكلاسيكية هي الطرق القديمة التي استخدمت فيما مضى من قبل اختراع الحواسيب ، وبقيت الأساس لكثير من الخوارزميات الحديثة التي تستخدم اليوم ، حيث كانت تعتمد على أحلال أو أبدال حرف مكان أخر ، والخوارزميات الجيدة كانت تقوم بالاثنين ، لكن جميع هذه الخوارزميات تعمل على الحروف فقط **Character-Based** أي على مدى 26 حرف ، بعكس الطرق الحديثة التي تعتمد على التعامل مع الBit (0 أو 1) ، وتقسم الطرق الكلاسيكية إلى قسمين رئيسين :

شفرات الإحلال Substitution Cipher :
في هذا النوع من الشفرات ، التشفير يكون عن طريق إحلال حرف من النص الأصلي Plaintext بحرف أخر ليكون هو الحرف المشفر cipher char ، عمليه الإحلال هذه تكون طريق جمع مفتاح ما إلى الحرف من النص الأصلي .

شفرات الإبدال Transposition :
في هذا النوع التشفير يكون عن تغيير أماكن حروف النص الأصلي ، أي مجرد تبديل في المواقع . (بعض الكتب تطلق على هذا النوع اسم **Permutation** تقليب) .

شفرات XOR :
هناك بعض الكتب تصنف طريقه التشفير هذه مع الطرق الكلاسيكية ، وبالرغم من ضعف هذه الطريقة ، إلا أن أغلب الشفرات الحديثة تعتمد على هذه العملية الرياضية في علميات أخرى (وقد تحدثنا عنها في الفصل الأول) .

نبدأ بالنوع الأول ، شفرات الإحلال :

تقسم شفرات الإحلال Substitution Cipher إلى أربعه أقسام رئيسيه :
النوع الأول : Monoalphabetic Substitution Cipher
النوع الثاني : Polyalphabetic Substitution Cipher
النوع الثالث : PolyGram Substitution Cipher
النوع الرابع : Homophonic Substitution Cipher

وسوف نتطرق لكل من هذه الطرق بالتفصيل ، أحب أن أنوه إلى أن هناك بعض الترجمات السيئة لهذه الأنواع ، لكني سأتحفظ عنها هنا ، وسنذكر المصطلح كما هو باللغة الأنجليزيه .

شفرات *Monoalphabetic Substitution Cipher* :

هذا النوع يعتبر من أقدم أنواع التشفير استخداما ، حيث نقوم في هذا النوع بإحلال Substitution حرف من النص الأصلي بحرف أخر جديد . وهو بالاضافه إلى قدمه يعتبر من أضعف أنواع التشفير ويسهل كسره باستخدام طريقه تسمى التحليل الإحصائي frequency analysis ، وهذه الطريقة من اكتشاف العالم العربي المسلم **أبو يعقوب الكندي** وهو أول من وضع أساسيات كسر الشفرات Cryptanalysis، حيث لاحظ وجود حروف تتكرر في القران الكريم أكثر من غيرها.

من أشهر شفرات هذا النوع Monoalphabetic Substitution :
Caesar Cipher
Affine Cipher
ROT13 Cipher
Abash Cipher

شفره قيصر *Caesar Cipher* :

من أحد أشهر أنواع التشفير الكلاسيكي ، حيث تتميز ببساطتها ويعيبها سهوله كسر الشفرة الناتجة ببساطه ،

وطريقه التشفير بأن نأخذ الحرف الأول من النص الأصلي ثم نقوم بجمع مفتاح (وهو دائما يكون 3 في شفره قيصر) مع النص الأصلي ، ويكون هو الحرف الأول في النص المشفر . وهكذا بالنسبة لباقي الحروف.

وفي حال كان الحرف هو الحرف الأخير في الأبجدية نقوم بالرجوع إلي بداية الحروف (تكون على شكل دائرة) .

انظر الصوره المقابلة :

Plaintext letter	A	B	C	D	……	W	X	Y	Z
Ciphertext letter	D	E	F	G	……	Z	A	B	C

نأخذ مثال لتوضيح هذا النوع من الشفرات:

النص الأصلي :
FIRE MISSILE

نبدأ بعمليه التشفير ، وكما ذكرنا كل حرف في النص الأصلي + المفتاح (3) ، وفي حاله تعدى الحرف Z نرجع من الأول .

الآن نقوم بأخذ الحرف الأول ونضيف إليه 3 حروف (الازاحه بمقدار 3 أحرف) ليكون:
F+3 = I
I+3 = L
ونكمل هكذا ، لباقي الحروف في النص الأصلي .

النص المشفر هو:
ILUH PLVVLOH

ولأن وضع الحروف المشفرة على نفس ترتيب الحروف في النص الأصلي يسهل من عمليه تخمين الكلمة ، نقوم بوضع النص المشفر على شكل block أو مجموعات كل منها يتكون من 5 حروف (جرت العادة على ذلك ، لكن بالطبع يمكنك تغيرها) .

الآن بعد وضع النص المشفر في شكل مجموعات كل منها يتكون من خمسه حروف يكون الناتج:
ILUHP LVVLO H

وهكذا أصبح النص أكثر تعقيدا لكاسر الشفرة ، ولكنها تبقي خوارزمية قيصر ضعيفة للغاية ، كما سنرى بعد قليل .

العملية العكسية ، وهي فك التشفير ،هنا كل ما علينا هو طرح ثلاثة حروف من كل حرف في النص المشفر ، ليخرج إلينا النص الأصلي.

اذا نستنتج أن لكل خوارزمية تشفير مفتاح معين ، هذا المفتاح (في الطرق التقليدية ، التي هي في الأصل تندرج تحت خوارزميات التشفير بالمفتاح المتناظر Symmetric Key Cryptography) يستخدم للتشفير ولفك التشفير ولذلك يجب أن يحفظ بمكان أمن . وفي حاله شفره قيصر ، مفتاح التشفير هو 3 (أزاحه Shift بمقدار 3) ، بالطبع يمكن استخدام أي مفتاح أخر ، لكنها لن تكون شفره قيصر .

كسر شفرة قيصر عن طريق التحليل الإحصائي *FREQUENCY ANALYSIS*

جميع اللغات (سواء عربيه أم انجليزيه أم أي لغة أخرى) تحتوي على حروف تتكرر دائما وباستمرار في الجمل ، في اللغة العربية على سبيل المثال الحرف أ ، ل .. الخ ، أما في اللغة الإنجليزية فالحرف E هو الحرف الأكثر تكرارا في الجمل.

الآن طريقه التحليل الإحصائي تعتمد على هذا الاحتمال ، فنقوم بالنظر في النص المشفر ونلاحظ الحرف الأكثر ترددا في النص ، بعدها قد يكون هذا الحرف في النص المشفر هو الحرف الأكثر تكررا في الجمل وهو E في اللغة الإنجليزية.

طبعا الاحتمال في الكثير من الأحيان يكون غير صحيح تماما ، ولكن يصلح في حال الطرق الضعيفة مثل شفره قيصر السابقة.

مثال ، لدينا الشفرة التالية ، ونريد كسرها وإرجاعها إلى حالتها الأصلية :

WFIDZ JVORT KCPVD GKZEV JJVDG KZEVJ JVORT KCPWF IDJFZ KZJNZ KYJVE
JRKZF EGVIT VGKZF EDVEK RCIVR TKZFE REUTF EJTZF LJEVJ JRCCK YZEXJ
RIVVJ JVEKZ RCCPV DGKPE FKSFI EEFKU VJKIF PVUEF KJKRZ EVUEF KGLIV
NZKYF LKCFJ JNZKY FLKXR ZEKYV IVWFI VZEVD GKZEV JJKYV IVZJE FWFID
EFJVE JRKZF EGVIT VGKZF EDVEK RCIVR TKZFE FITFE JTZFL JEVJJ EFVPV
VRIEF JVKFE XLVSF UPDZE UEFTF CFIJF LEUJD VCCKR JKVKF LTYFS AVTKF
WKYFL XYKEF JVVZE XREUJ FFEKF EFKYZ EBZEX EFZXE FIRET VREUE FVEUK
FZXEF IRETV EFFCU RXVRE UUVRK YEFVE UKFFC URXVR EUUVR KYEFR EXLZJ
YTRLJ VFWRE XLZJY TVJJR KZFEG RKYEF NZJUF DREUE FRKKR ZEDVE KJZET
VKYVI VZJEF KYZEX KFRKK RZEKY VSFUY ZJRKK MRCZM VJKYL JNZKY EFYZE
UIRET VFWDZ EUEFY ZEUIR ETVRE UYVET VEFWV RIWRI SVPFE UUVCL UVUKY
FLXYK IZXYK YVIVZ JEZIM RER

الخطوة الأولى هي معرفه الحرف الأكثر تكرارا ، ونظرا لطول الشفرة فيفضل عد كل حرف يتكرر ،

نبدأ بالعد ، نقوم بعد الحرف الأول وهو W ونلاحظ كم مره تكرر واحد ، اثنين ، ثلاثة ..... إلى أن نصل إلى نهاية الشفرة لنعرف أن الحرف W تكرر 9 مرات ، نأخذ الحرف الثاني وهو F ونبدأ بالعد ، وهكذا مع باقي الحروف في الشفرة.

نتيجة تكرار الحروف بعد العد:

A: 1 B: 1 C: 16 D: 14 E: 82 F: 69 G: 10 H: 0 I: 27 J: 47 K: 61
L: 15 M: 3 N: 5 O: 2 P: 8 Q: 0 R: 45 S: 5 T: 21 U: 28 V: 69
W: 9 X: 15 Y: 28 Z: 47

نلاحظ في النتيجة أعلاه ، أن الحرف E هو الحرف الأكثر تكرارا في النص المشفر (تكرر 82 مره) ، الآن كما ذكرنا سابقا الحرف الأكثر تكرار في الشفرة قد يكون هو الحرف E ، ولأن في حالتنا هذه ، الحرف المشفر هو E اذا بالتأكيد الحرف E لن يكون هو الحرف البديل ، لذلك سوف نأخذ الحرف الأكثر تكرارا في الشفرة أتي بعد الحرف E .

الآن لدينا حرفين هما F,V حيث تكرر كل منهم 69 مره ، وقد يكون أحدهم هو الحرف E .

الآن نأخذ الحرف F ونشاهد الفرق بينه وبين E ، والنتيجة هي 1 (لأن الحرف F يأتي بعد E) . أيضا نأخذ الحرف الثاني هو V ونشاهد الفرق بينه وبين الحرف E والنتيجة هي 17 . (الحرف Vيأتي بعد 17 حرف من E) .

إلى هنا ، أصبح لدينا احتمالين ،

الأول هو أن يكون المفتاح هو 1 ، والثاني أن يكون المفتاح هو 17.

نقوم بتجربة الأول ، ونطرح من كل حرف في الشفرة 1 (نرجع إلى الوراء بمقدار حرف) ، النتيجة ستصبح غير مفهومه وبالتالي الازاحه بمقدار 1 خاطئه.

نجرب الازاحه بمقدار 17 ، وسوف نشاهد هذا النص الجميل:

```
FORMI SEXAC TLYEM PTINE SSEMP TINES SEXAC TLYFO RMSOI TISWI THSEN
SATIO NPERC EPTIO NMENT ALREA CTION ANDCO NSCIO USNES SALLT HINGS
AREES SENTI ALLYE MPTYN OTBOR NNOTD ESTRO YEDNO TSTAI NEDNO TPURE
WITHO UTLOS SWITH OUTGA INTHE REFOR EINEM PTINE SSTHE REISN OFORM
NOSEN SATIO NPERC EPTIO NMENT ALREA CTION ORCON SCIOU SNESS NOEYE
EARNO SETON GUEBO DYMIN DNOCO LORSO UNDSM ELLTA STETO UCHOB JECTO
FTHOU GHTNO SEEIN GANDS OONTO NOTHI NKING NOIGN ORANC EANDN OENDT
OIGNO RANCE NOOLD AGEAN DDEAT HNOEN DTOOL DAGEA NDDEA THNOA NGUIS
HCAUS EOFAN GUISH CESSA TIONP ATHNO WISDO MANDN OATTA INMEN TSINC
ETHER EISNO THING TOATT AINTH EBODH ISATT VALIV ESTHU SWITH NOHIN
DRANC EOFMI NDNOH INDRA NCEAN DHENC ENOFE ARFAR BEYON DDELU DEDTH
OUGHT RIGHT HEREI SNIRV ANA
```

هناك احتمال كبير ، أن يصعب عليك ترتيب الحروف السابقة وإرجاعها إلى حالتها الأصلية بسبب عدم إتقان اللغة الأنجليزيه بشكل جيد . لكنها تبقى في النهاية هي النص الأصلي ، ويفضل في تلك الحالة الإستعانه بأحد المترجمات مثل الوافي والبدء في محاوله تجميع الحروف وترجمتها .

نعود إلى الطريقة Monoalphabetic حيث أنه ممكن اختيار **جمله للتشفير Key Phrase** ، بدلا من المفتاح (الازاحه) .

مثلا ، لدي جمله التشفير Key Phrase هذه:
THE HILLS ARE ALIVE

اذا أردت أن أشفر الحرف A بهذه الطريقه ، سوف يكون الحرف A بعد التشفير هو الحرف Tلأنها الأولى في جمله التشفير .
و B يصبح H ، و C يصبح E ، وهكذا...

وفي حال انتهت جمله التشفير ولم ينتهي النص الذي أريد تشفيره ، فأقوم بتكملة الحروف بالحروف العادية ، **لمزيد من التوضيح انظر الجدول في الصفحة التالية :**

مثال (صغير) لتوضيح الطريقة ،
لدى جمله التشفير التالية : WAJDY
وأريد تشفير العبارة : STOP FIRE
نقوم أولا بوضع النص الأصلي ومن أسفله النص المشفر بالاضافه إلى باقي الحروف :
Plaintext : ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cipher text : WAJDYBCEFGHIKLMNOPQRSTUVXZ

Plaintext Letter	Ciphertext Letter
A	T
B	H
C	E
D	I
E	L
F	S
G	A
H	R
I	V
J	B
K	C
L	D
M	F
N	G
O	J
P	K
Q	M
R	N
S	O
T	P
U	Q
V	U
W	W
X	X
Y	Y
Z	Z

الآن أقوم بتشفير النص STOP FIRE ، وينتج لدى النص المشفر التالي : QRMN BFPY ، وأقوم بتقسيم النص المشفر إلى Block يتكون من خمسه حروف ، لينتج لدي في النهاية QRMNB FPY .

شفره أتباش *Atbash Cipher*

هذه الشفرة أيضا من أبسط أنواع الشفرات ، وهي كانت في الأصل للغة العبرية ، ولكن يمكن استخدام المفهوم في باقي اللغات .
وطريقتها كالتالي ، وهي أن نجعل الحرف الأول في اللغة هو الحرف الأخير ، والحرف الثاني هو قبل الأخير ، وهكذا...

Plain: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cipher: ZYXWVUTSRQPONMLKJIHGFEDCBA
مثلا ، لتشفير الكلمة money ، يصبح لدينا الناتج nlmvb .

في بعض الأحيان من الممكن أن الكلمات بعد التشفير يكون لديها معني وهي مشفره مثل:
النص الأصلي = "hob" بعد التشفير تصبح "sly" ، وهكذا للكلمات الأخرى
النص الأصلي = "hold" بعد التشفير تصبح "slow"
النص الأصلي = "holy" بعد التشفير تصبح"slob"
"horn" = "slim"
"irk" = "rip"
"low" = "old"
"glow" = "told"
"grog" = "tilt".

ومع ذلك تبقى تلك الشفرات من أسهل الأنواع على الإطلاق!

شفره *ROT13*

تعتبر هذه الشفرة (كما هو الحال مع جميع شفرات نوع Monoalphabetic) ضعيفة للغاية ، حيث أن التشفير وفك التشفير يتم بنفس الطريقة ، و مفتاح التشفير 13 ، وللتشفير نقوم بجمع 13 على الحرف الأول من النص الأصلي ، ولفك التشفير تقوم أيضا بجمع 13 على الحرف الأول من النص المشفر .

P = ROT13 (ROT13 (P))

الحرف p يعني الحرف الأول من النص الأصلي Plaintext ، نقوم بعدها بتشفيره بجمع 13 حرف إليه، لنفرض أن الحرف الأول من النص الأصلي هو D ، الحرف D قيمته 3 ، نجمع (13+3)%26 والناتج هو 16 ، أو ممكن نتحرك 13 خطوه من الحرف D والناتج في النهاية سواء بالجمع أو بالتحرك هو الحرف Q .

قبل أن نبدأ عمليه التشفير دائما ، نضع هذا الجدول الذي سنستخدمه كثيرا لتسهيل معرفه مواقع الحروف:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

عمليه فك التشفير ، تكون أيضا بجمع 13 إلى الحرف المشفر (أو التحرك 13 خطوه إلى الأمام) . نجرب على الحرف Q الذي كان نتيجة عمليه التشفير السابقة ، 3 = 26%(16+13) ، أو نتحرك 13 خطوه إلى الأمام حتى نصل إلى الحرف الأصلي وهو D .

شفره ROT13 استخدمت في الأصل في برنامج كان في نظام Unix ، وحاليا لا يوجد أي استخدام لها ، ما عدا في بعض المسابقات **Puzzle** ويكون الجواب موجود (مخفي) بهذا الشكل .

شفره ***Affine Cipher***

التشفير بطريقة **Affine Cipher** (البعض يترجمها بطريقه **التشفير المختلط** ، لأنه هنا خلط بين نوعين من التشفير ، النوع الأول وهو شفره قيصر ، والأخر وهو شفره الضرب product Cipher)

في شفره قيصر يكون التشفير كالتالي :

c = p + key MOD n

(وهنا key يعتبر الازاحه) .

وفي شفره الضرب ، يكون التشفير كالتالي :

C = p * key MOD n

الآن في شفره Affine (أو الشفره المختلطة) جمعت بين الطريقين ، حيث يتم الجمع والضرب أيضا .

C = m*p + key MOD n

لكن هناك شرط مهم جدا ، وهو أن تكون m و n هما أوليان فيما بينهما ، أي أن القاسم المشترك الأعظم ل m و n يساوي 1 . وفي حال لم ينفذ هذا الشرط ، فانه لن يمكن فك التشفير. (راجع الفصل الأول ، لمعرفه المزيد عن القاسم المشترك الأعظم وكيفيه إيجاده) .

الآن لفك التشفير يجب أن نوجد معكوس m (في حال كان GCD(m,n) = 1 فإنه يمكن أيجاد ذلك المعكوس ، لذلك كما ذكرنا قبل قليل يجب تحقيق الشرط منذ البداية ، حتى يكون هناك المعكوس).

لفك التشفير ، نتبع :

P = m` * (c – key) (MOD n)

<u>نأخذ مثال ليوضح العملية :</u>

نريد تشفير العبارة : WAR LOST
والمفتاح key يساوي 10 ، ومعامل الضرب m يساوي 7 .

قبل أن نبدأ في التشفير ، يجب أن نتأكد من أن هناك معكوس ل m ، حتى يمكن فك الشفره .
نأخذ القاسم المشترك الأعظم ل m و n (26 لأنها عدد الحروف) .
GCD(7,26) = 1 ، ويساوي واحد ، اذا هكذا نتأكد من أن هناك معكوس ل M .

نضع جدول الحروف ، حتى يساعدنا في معرفه موقع الحروف :

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

الآن بعد تحويل النص الأصلي إلى أرقام ، يكون بهذا الشكل :
22 0 17 11 14 18 19

الآن نبدأ في التطبيق في القانون :
C = m * p + key MOD 26

C1 = 7 * 22 + 10 MOD 26 = 8
C2 = 7 * 0 + 10 MOD 26 = 10
C3 = 7 * 17 + 10 MOD 26 = 25
C4 = 7 * 11 + 10 MOD 26 = 9
C5 = 7 * 14 + 10 MOD 26 = 4
C6 = 7 * 18 + 10 MOD 26 = 6
C7 = 7 * 19 + 10 MOD 26 = 13

الآن النتيجة بعد التشفير هي :
8 10 25 9 4 6 13
ونقوم بتحويلها إلى حروف ، ليصبح لدينا : IKZJE GN

الآن لفك التشفير ، يجب أن نعرف ما هو معكوس m ، حتى نستطيع التطبيق في القانون التالي :
P = m` * (c – key) (MOD 26)

كيف يمكن إيجاد المعكوس ، وذلك عن طريق خوارزمية اقليدس الممتدة (التي ذكرناها في الفصل الأول) .

ندخل العددين 7 و 26 في خوارزمية اقليدس الاقليدية الممتدة ، لينتج لدينا المعكوس : 15

الآن نطبق في القانون :

P1 = 15 * (8 – 10) MOD 26 = 22
P2 = 15 * (10 – 10) MOD 26 = 0

P3 = 15 * (25 – 10) MOD 26 = 17
P4 = 15 * (9 – 10) MOD 26 = 11
P5 = 15 * (4 – 10) MOD 26 = 14
P6 = 15 * (6 – 10) MOD 26 = 18
P7 = 15 * (13 – 10) MOD 26 = 19

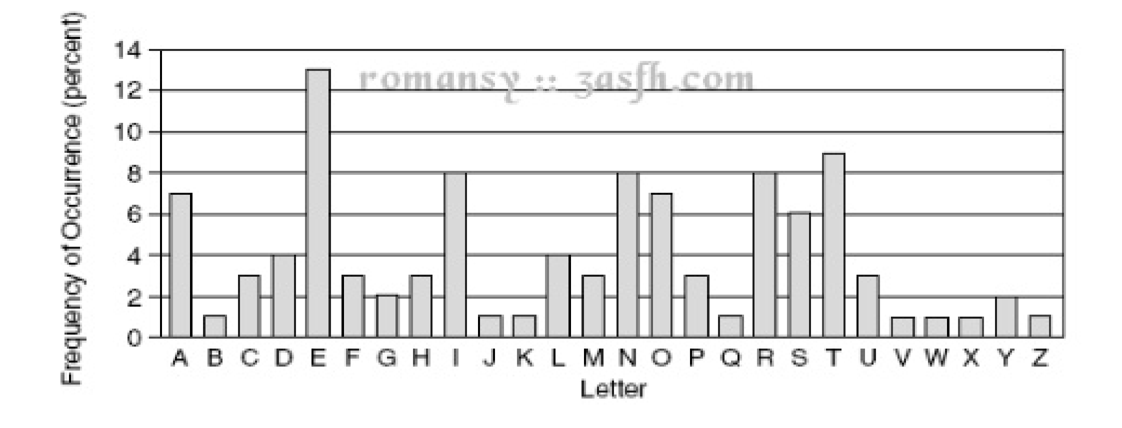
اذا النص الأصلي هو : 19 18 14 11 17 0 22 ، وبعد تحويله إلى حروف يصبح :
WARLO ST (وهو المطلوب) .

كسر الشفرات من نوع *MONOALPHABETIC*

طريقه التحليل الإحصائي هي الطريقه الشائعة لمثل هذه الطرق (حيث تستبدل هذه الطرق كل حرف في الشفرة الاصليه بحرف ما اعتمادا على أزاحه معينه -مفتاح ما- أو جمله للتشفير) .
حيث تستفيد من أن بعض الحروف في كل لغة معدل تكرارها أكثر من باقي اللغات.

وطبعا للبدء في هذه الطريقه ، يجب أن يكون هناك نص كافي للبدء في مرحله العد ، يعني نص يتكون من 10 حروف أو أقل يصعب جدا كسره (إن لم يكن مستحيل في الأصل) بهذه الطريقه.

الجدول التالى بين معدل تكرارات الحروف فى اللغة الأنجليزيه:



تمرين ، قم بفك الشفرة التالية ، علما بأنها مشفره بخوارزمية قيصر بمفتاح مختلف:
fqjcb rwjwj vnjax bnkhj whxcq nawjv nfxdu mbvnu ujbbf nnc

الحل :
في البداية أول خطوه هي معرفه كم مره تكرر كل حرف ، نستطيع أن نعرف ما هو الحرف الذي تكرر أكثر من غيره ، ومنه قد يكون هو الحرف E .

الآن نبدأ في عد الحروف:
a:2 , b:5 , c:3 , d:0 , e:0 , f:3 , g:0 , h:2 , i:0 , j:7 , k:1 , l:0 , m:1 , n :7
وهكذا لباقي الحروف
(كل الباقي لا يتعدى ثلاثة أحرف ، ما عدا الحرف w تكرر أربعه مرات) .

نعود إلى الحروف الأكثر تكرار وهي j و n حيث تكرر كل منهم 7 مرات.

الآن نشاهد الفرق بين الحرف الأول j والحرف E ليخرج الناتج 5 ، الآن المفتاح هنا 5 ، ونبدأ بفك الشفرة:
نطرح من كل حرف 5 حروف:
f - 5 = a
q - 5 = l
j - 5 = e
c - 5 = x
b - 5 = w
والناتج يكون:
alexw mrere ajevs نتوقف هنا بالطبع ، لأن النص ليس له معنى بتاتا.
نأخذ الحرف الثاني الأكثر تكرارا ، وهو الحرف N .
نشاهد الفرق بينه وبين الـ E، e-n=9 ، اذا المفتاح في هذه الحالة يكون 9 .
الآن نطرح من كل حرف 9 حروف....

f-9 = w
q-9 = h
j-9 = a
c-9 = t
b-9 = s

وهكذا ليخرج لدينا:
whats inana mearo sebya nyoth ernam ewoul dsmel lassw eet

الآن نبدأ بمعرفه الجمل ، وبعد قليل من المحاولات والاستعانة بالمترجم الوافي ، يخرج لدينا
what's in a name a rose by any other name would smell as sweet

تمرين : قم بفك الشفرة التالية ، واستخدم الجملة التالية لفك التشفير"monalphbetic"
jmjmj gsmsg lrjgu csqyj quflr mfajq erdmc cmqlv lqyhg gawgq arpgq sblce jrlrj lnmec cyjqu flrmf ajqer d

هذا التشفير عن طريقه جمله التشفير سهل للغاية ، كل ما في الأمر ، هو وضع الحروف العادية ووضع جمله التشفير أسفل منها ، وبعدها :

a b c d e f g h I j k l m n o p q r s t u v w x y z
m o n a l p h b e t I c d f g j k q r s u v w x y z

الآن نأخذ الحرف الأول من النص المشفر وهو الحرف j و ننظر إلى ما يقابل الحرف j في السطر الأول الذي يمثل النص الأصلي ، ليصبح j هو الحرف p ، و m هو الحرف a ، وهكذا لباقي الحروف في هذه الشفرة .

ليخرج إلينا في النهاية النص الأصلي التالي:

Papap otato espou ltryp runes andpr ismar eallv erygo odwor dsfor theli psesp ecial lypru nesan dpris m

وبعد أعاده ترتيبها ـ ليصبح لدينا :

Papa potatoes poultry prunes and prism are all very good words for the lips especially prunes and prism

(السؤالين السابقين ، منقولين من منتديات الفريق العربي للبرمجة www.arabteam2000.com للأخ **MEMONONO**)

شفرات *Polyalphabetic substitution cipher*

الشفرات التي تندرج تحت هذا النوع ، تقوم هي مجرد تطبيق طريقه **Monoalphabetic** عليها عده مرات ، أي أن المفتاح هنا يكون عبارة عن عده مفاتيح . مثلا اذا كان عدد المفاتيح 4 ، يشفر الحرف الأول بالمفتاح الأول والحرف الثاني بالمفتاح الثاني ، وهكذا . وعندما تنتهي المفاتيح بعض الطرق تقوم بإعادته كتابته مره أخرى ، وبعضا لا تقوم ، كما سنذكرهم بعد قليل .

وأشهر الشفرات تحت هذا النوع ، هي شفرات **عائله فجينير Vigenere Cipher** وقد طورت هذه الشفرات على مدى السنين من قبل أناس مختلفين ، أيضا جهاز التشفير الألماني **Engima** يندرج تحت هذا النوع . ومن هذه الشفرات :

Simple Shift Vigenere Cipher
Full Vigenere Cipher
Auto-Key Vigenere Cipher
Running Key Vigenere Cipher

شفره *Simple Shift Vigenere Cipher*

طريقة التشفير في هذا النوع من أبسط ما يكون ، حيث نقوم بتشفير الحرف الأول بالمفتاح الأول ، والحرف الثاني بالمفتاح الثاني ، وهكذا .. وفي حال انتهت المفاتيح أقوم بتكرار كتابتها مرة أخرى .

مثال بسيط لتوضيح التشفير هذه الطريقه:

لدي هذه العبارة (النص الأصلي) :
DEFCON FOUR

أريد أن أشفرها بهذه الطريقه ، أول خطوه هي أن يكون المفتاح متغير ، أي مختلف من موقع لأخر.

مثلا قد يكون المفتاح على الشكل:
الحرف الأول في النص يشفر بالمفتاح : 5
الحرف الثاني يشفر بالمفتاح : 13
الحرف الثالث يشفر بالمفتاح : 2
الحرف الرابع يشفر بالمفتاح: 7

اذا المفاتيح (تسمى بـ **Key Length**) في هذه الحالة هي : 5 13 2 7

قبل أن نبدأ عمليه التشفير ، نضع هذا الجدول لتسهيل معرفه مواقع الحروف:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

الآن نضع النص المراد تشفيره ، ومقابله نضع المفتاح ، ومن ثم نبدأ بعمليه الازاحه ليخرج لدينا النص المشفر

نبدأ بالحرف الأول من النص الأصلي وهو D ، والمفتاح الأول هو 5 ،
الحرف المشفر الأول D + 5 = ويساوي I (أو ممكن نأخذ قيمه الD وهي 3 ونجمع إليها 5 مع أخذ % 26 ليخرج لدينا الناتج وهو 9 ، الذي يمثل الحرف I) .
نأخذ الحرف الثاني ، وهو E ، والمفتاح الثاني وهو 13 ، وبعد عمليه الإزاحة ينتج لدينا الحرف المشفر R
وهكذا لباقي الحروف في النص....

هذه الصوره توضح عمليه التشفير ، السطر الأول هو النص الأصلي ، السطر الثاني المفاتيح ، وفي حال انتهت نعيد كتابتها مره أخرى ، السطر الأخير هو الناتج من جمع السطر الأول مع الثاني وهو النص المشفر:

Plaintext	D	E	F	C	O	N	F	O	U	R
Shift value	5	13	2	7	5	13	2	7	5	13
Ciphertext	I	R	H	J	T	A	H	V	Z	E

الآن نأخذ الناتج ونضعه في شكل Block كل منها يتكون من 5 حروف:
اذا النص المشفر هو:
IRHJT AHVZE

إلى هنا الأمر بسيط للغاية ، ولكن تبقى مشكله فعليه وهي صعوبة تذكر المفتاح وخاصة اذا كان طويل ، فكيف أحفظ هذه المفاتيح ؟ الحل هو استخدام نص أو جمله بدل هذه الأرقام ، وعند التشفير أعوض بكل حرف من هذه الجملة بالرقم على حسب موقعها ، مثلا الحرف A المفتاح هنا هو 0 ، الحرف B المفتاح هو 1 وهكذا...

اذا لو لدينا المفتاح (المفاتيح) : 1 12 0 18 19 14 5 5

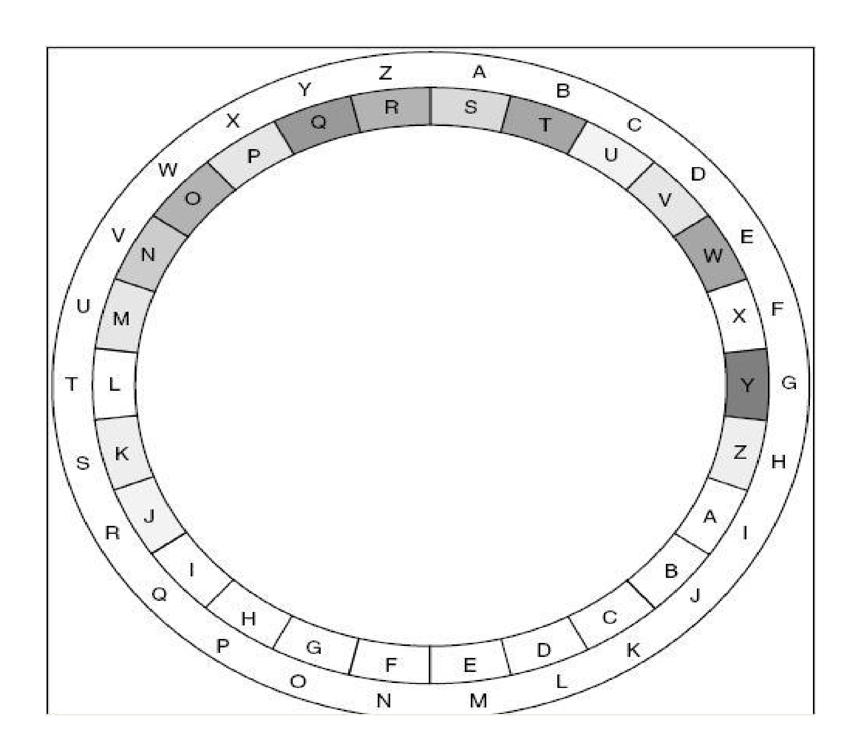
سوف يكون بالشكل التالي ، بعد تعوضه بالحروف : BLAST OFF

وهكذا ، سوف نقضي على مشكله حفظ المفاتيح الطويلة ، عن طريق جمله التشفير .

أحد أشهر الاستخدامات لهذا النوع من الشفرات هو عجله الأكواد Code Wheel :

حيث يكون لدينا عجله (دائرة متحركة) تتكون من جزئيين داخلي وخارجي ، الجزء الداخلي فيه الحروف الأبجدية وهو يمثل الحروف المشفرة ، والجزء الخارجي أيضا يحتوي على الحروف الأبجدية ولكنه يمثل الحروف في النص الأصلي ،

اذا كنت لا تستطيع تخيل شكل هذه العجلة -وهو ما حصل بالتأكيد- أنظر الصوره المقابلة :



وهذه صوره لها وهي في العالم الحقيقي :



كيف نقوم بالتشفير في هذه العجلة ، ببساطه أولا أقوم باختيار المفتاح وهو في هذه الحالة عبارة عن حرف واحد ، وليكن اخترت المفتاح S .

الآن قبل أن أبدأ بالتشفير ، أحرك الجزء الداخلي (الخاص بالشفرة) وأجعل الحرف S تحت الحرف A كما هو موضح في الصوره أعلاه (الصوره الأولى للعجلة) .

الآن اذا أردت أن أشفر الحرف G ، أنظر في الجزء الخارجي إلى G ، بعدها أشاهد ماذا يقابله وهو الحرف. Y وهكذا لباقي الحروف في النص.

فك التشفير هو العملية العكسية ، أريد أن أفك تشفير الحرف H ، أنظر في الجزء الداخلي إلى H وأنظر ما يقابله وهو الحرف p . وهكذا...

كسر شفرات فجينير *Vigenere* البسيطة

لأن المفتاح يتكرر في هذه الشفرات بحيث تعيد دوره المفاتيح من البداية من مره (تعرف هذه بالـ **Period** أي الفترة التي ما بين تكرار المفتاح ، وكلما كانت هذه الفترة أطول، كلما كانت الشفرة أكثر أمانا)، يمكن استغلال هذه الإعادة وتطبيق طريقه التحليل الإحصائي ، ولكن يجب أن يكون النص المشفر كبير بما يكفي ، أيضا يجب أن يتم معرفه عدد المفاتيح المستخدمة أو **Key Length** وهو أصعب جزء في العملية.

وفي حال تم معرفه عدد المفاتيح Key Length نقوم بتقسيم النص المشفر إلى مجموعه صغيره من النصوص ، بعدها وبطريقه ما نطبق عليها طريقه التحليل الإحصائي على كل مجموعه على حده ، ونلاحظ ما هو الحرف الذي يتكرر كثيرا في المجموعة الأولى وقد يكون هو E (نعيد نفس خطوات كسر Monoalphabetic)، طبعا على كل مجموعه من المجموعات الصغيرة.

مثال ، لدينا النص المشفر الأتي:
XZQTY IISTN PAWRT GSGPO LNOXF

اذا قمنا بتخمين المفاتيح مثلا 5 مفاتيح ، وكانت هي الإجابة الصحيحة ، الآن نقوم بتقسيم النص إلى مجموعه من النصوص ، ويتم التقسيم وذلك بأخذ حرف واحد من كل بلوك Block في النص المشفر.

نأخذ الحرف الأول من البلوك الأول وهو X
نأخذ الحرف الأول من البلوك الثاني وهو I
نأخذ الحرف الأول من البلوك الثالث وهو P
نأخذ الحرف الأول من البلوك الرابع وهو G
نأخذ الحرف الأول من البلوك الخامس وهو L
وهكذا نستمر بأخذ الحرف رقم 1 من أي بلوك n ، إلى أن تنتهي جميع البلوك

اذا المجموعة الأول هي XIPGL ، ونطبق نفس الطريقه على الحرف الثاني من جميع البلوكات ، والثالث والرابع والخامس....

وبعد ترتيبها في شكل مجموعات ، **يتكون لدينا:**

Key Length = 5				
Category 1	Category 2	Category 3	Category 4	Category 5
XIPGL	ZIASN	QSWGO	TTRPX	YNTOF

السؤال الذي يطرح نفسه كيف تمت معرفه طول المفتاح (عدد المفاتيح) ؟ ، طبعا في المثال السابق لأن النص كان صغير لن نستطيع إيجاده ، ولكن الطريقه صحيحة ، لذلك نأخذ مثال أخر بعد قليل لتوضيح الأمر ، مجددا.

(التخمين العشوائي يفيد كثيرا ، لكنه يستلزم الكثير من المحاولات ، حتى تستطيع معرفته عشوائيا) .

طريقه كيسسكي *KAISISKI* لمعرفه عدد المفاتيح *Key Length*

هذه الطريقه بسيطة للغاية ، حيث علينا هنا ملاحظه حرفين أو ثلاثة حروف يتكرروا كثيرا ولكن بشرط أن يأتوا مع بعض ، يعني مثلا تكررت الحروف XYZ كثيرا ، وكل مره تتكرر فيها هذه الحروف تأتي مجمتعه مع بعض بهذا الشكل ، هنا في هذه الطريقه حتى نعرف عدد المفاتيح نقوم بعد الحروف التي تأتي بين الـ XYZ الأولى والثانية ، ونقوم بعد الحروف بين الـ XYZ الثانية والثالثة ، وهكذا إلى أن ننتهي ، بعدها يمكن بقليل من المحاولة معرفه عدد المفاتيح.

مثال بسيط ،

لنفترض أن لدينا شفره ما ، وفيها الحروف FSI تتكرر كثيرا ، والفرق بين كل FSI في هذا النص المشفر موضحه في الجدول التالي:

I	Distance between (I–1)th and Ith occurrence
2	56
3	14
4	35
5	63
6	9
7	5
8	28
9	35
10	33
11	21
12	35

-طريقه العد بأن نعد من بعد الحرف F إلى أن نصل إلى الحرف F في ال FSI الثانية -

لو دققنا كثيرا في هذه الأعداد ، سنعرف أن أغلبها يكون من مضاعفات العدد 7 ، وهناك 3 أعداد في هذا الجدول ليست من مضاعفات السبعة ، ولكن قد يكون الحرف F ظهر لوحده ، وليس مجتمع مع الـ FSI ، وتم عده.

مثال شامل لفك التشفير بهذه الطريقه:

ليكن لدينا الشفرة التالية ، قم بكسر الشفرة ومعرفه عدد المفاتيح المستخدمة ، علما بأنها مشفره بطريقه فجينير البسيطة:

LJVBQ STNEZ LQMED LJVMA MPKAU FAVAT LJVDA YYVNF JQLNP LJVHK VTRNF LJVCM LKETA LJVHU YJVSF KRFTT WEFUX VHZNP

الخطوة الأولى هي أن نعرف عدد المفاتيح المستخدمة Key Length ، لذلك نستخدم طريقه كسيسكي ، ونبدأ في ملاحظه حروف مجتمعه تأتي أكثر من غيرها.

دقق كثيرا في الشفرة ، سوف تلاحظ أن الحروف LJV تأتي مجتمعه بالاضافه إلى أنها تتكرر. نبدأ العد بين كل LJV وأخرى في هذه الشفرة ،

الفرق بين أول حرف L والحرف L في البلوك الثاني هو 15
والفرق بين ثاني حرف L والحرف L في البلوك الثالث هو 15 أيضا.
ونقوم بالعد هكذا إلى أن نصل إلى النهاية.

وهذا الجدول يوضح الفرق بين كل حرف L في النص:

Occurrence	Distance
2	15
3	15
4	15
5	10
6	10

الآن نأتي إلى مرحله معرفه عدد المفاتيح ، لاحظ الأرقام في الجدول 10,15,15 ، احتمال تكون المفاتيح (وهو الاحتمال الصحيح) هو 5 مفاتيح.
الآن بعد معرفه طول المفاتيح ، يجب معرفه عدد هذه المفاتيح ، نبدأ بتقسيم الشفرة إلى 5 أقسام ، ونأخذ الحرف الأول من البلوك الأول والحرف الأول من البلوك الثاني ، ونفس الكلام بالنسبة للرابع والخامس والسادس إلى نهاية الشفرة، ونضعهم في القسم الأول.

وأيضا للقسم الثاني ، نأخذ الحرف الثاني من كل بلوك إلى نهاية الشفرة،
ونفس الكلام لباقي الأقسام.

الآن بعد ترتيب في 5 أقسام نطبق الطرق السابقة Monoalphabetic على كل قسم على حده ، وهنا نريد أن نعرف ما هو الحرف الأكثر تكرارا في كل قسم.

الشكل التالي يبين شكل الأقسام الخمسة ، وما هو الحرف الأكثر تكرارا في كل قسم:

Category	Letters	Most Common Letter
1	LSLLM FLYJL VLLLY KWV	L
2	JTQJP AJYQJ TJKJJ REH	J
3	VNMVK VVVLV RVEVV FFZ	V
4	BEEMA ADNNH NCTHS TUN	N
5	QZDAU TAFPK FMAUF TXP	A

نبدأ بالقسم الأول:
LSLLM FLYJL VLLLY KWV

الحرف الأكثر تكرارا هو الـ L ،

السؤال الذي يطرح نفسه ، ما هو الحرف المحتمل أن يكون E أو T أو R أو غير ذلك ؟

لاحظ هنا أن النص صغير جدا ، ولذلك سوف يكلفنا الكثير من المحاولات اذا أردنا أن نعتبر أن Eمثلا هو الحرف الذي يتشفر إلى L ، لأننا سوف ننتقل إلى القسم الثاني ونكرر نفس الطريقه في الحرف الأكثر تكرارا وهو الحرف J وقد يكون هو E وليس الحرف السابق !

لذلك اذا أردنا أن نجرب هكذا سوف نخرج بنتيجة في النهاية ولكن قد يطول الأمر كثيرا ،

لذلك نبحث عن حل أخر ، وبقليل من الملاحظة حول الأحرف الثلاثة التي تأتي دائما مع بعض ، ونحن نعرف في اللغة الأنجليزيه أن THE تتكون من ثلاثة أحرف ،

لذلك قد تكون L في القسم الأول هي T
وتكون j في القسم الثاني هو الحرف H
وتكون V في القسم الثالث هو الحرف E .

الآن نأخذ الفرق بين هذه الحروف أي بين الـ T والـ L والناتج هو 18 (الحرف S)
الفرق بين الحرف H والحرف J هو 2 (الحرف C).
الفرق بين الحرف E والحرف V هو 17 (الحرف R) .

إلى هذه المرحلة ، عرفنا أول ثلاثة مفاتيح في هذه الشفرة SCR وتبقى اثنان

نعود إلى القسم الرابع والخامس ، حيث القسم الرابع الحرف الأكثر تكرارا هو N والقسم الخامس الحرف الأكثر تكرارا هو A . وهنا تبدأ مرحله التخمين والبحث عن هذين الحرفين (المفتاحين) ، يمكنك في البداية اعتبار كل منهم حرف ما وليكنA,B .

الآن نقوم بفك الشفرة على أساس المفتاح هو SCRAB لتخرج لدينا شفره نصفها صحيح والنصف الأخر خاطئه ، ونقوم بعدها بتجربة حرفين آخرين ، وهناك حوالي 26*26 احتمال يجب البحث فيه حتى نطلع الحرفين الأخيرين ، وهو أمر شاق ومتعب يدويا ، لكن باستخدام الحاسب فالأمر سهل وسريع .

إلى أن نصل أخيرا إلى المفتاحين الصحيحين الذين هما A,M ولتكون جمله التشفير هي SCRAM .

Category	Letters				Most Common Letter
1	LSLLM	FLYJL	VLLLY	KWV	L
2	JTQJP	AJYQJ	TJKJJ	REH	J
3	VNMVK	VVVLV	RVEVV	FFZ	V
4	BEEMA	ADNNH	NCTHS	TUN	N
5	QZDAU	TAFPK	FMAUF	TXP	A

الآن نفك تشفير النص المشفر السابق بهذه الجملة SCRAM ، ليخرج لدينا:

THEBE ARWEN TOVER THEMO UNTAI NYEAH THEDO GWENT ROUND THEHY DRANT THECA TINTO THEHI GHEST SPOTH ECOUL DFIND

وبعد أعاده ترتيبها يصبح لدينا النص الأصلي هو:

THE BEAR WENT OVER THE MOUNTAIN YEAH THE DOG WENT ROUND THE HYDRANT THE CAT INTO THE HIGH EST SPOT HE COULD FIND

طريقه فجينير الكاملة *THE FULL VIGENERE CIPHER*

هنا في هذه الطريقه بعدما أختار الجملة (مفاتيح التشفير) ، يكون التشفير عن طريق أخذ الحرف الأول من النص الأصلي مع الحرف الأول من جمله التشفير وأخذ نقطه التقاطع في جدول التشفير **a tabular recta** ،

والجدول شكله كما هو واضح في الشكل التالي:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

الآن مثلا لدي النص التالي:
HARKONNEN RULZ

وأريد أن أشفره باستخدام هذه الجملة : SPICE

أقوم في البداية بأخذ نقطه التقاطع بين الحرف الأول في النص الأصلي والحرف الأول في الجملة .

جميع الأعمدة تمثل الحروف في النص الأصلي ، السطور تمثل الحروف في جمله التشفير.
نبدأ بأخذ الحرف H من النص ، والحرف S من الجملة

H و S يتقاطعا في الحرف O .
A و P يتقاطعا في الحرف Z .
وهكذا بالنسبة لباقي الحروف في النص الأصلي ، ونقوم بتكرار جملة التشفير في حال انتهت .

لينتج لدينا النص المشفر : OZTJYJTZGDKPX
وأقوم بفصلهم كل خمسه حروف على حده: OZTJY JTZGD KPX

ولفك التشفير ، نقوم بالعملية العكسية ، وفي هذه الحالة تكون الأعمدة هي النص الأصلي ، والسطور هو المفاتيح (نفس طريقة التشفير) ، ولكن لكي أفك تشفير حرف ما ، أولا أخذ الحرف الأول من جمله التشفير ، وأبحث في السطر الخاص بالحرف الأول من جملة التشفير عن الحرف المشفر الأول ، وعندما أجده يكون الحرف الموجود في العامود هو الحرف الأصلي .

مثال لتوضيح عمليه فك التشفير :

فك تشفير العبارة : OZTJY JTZGD KPX
عن طريق الجملة : SPICE

نبدأ بأخذ الحرف الأول من جملة التشفير وهو S ، نذهب إلى الجدول ، ونشاهد السطر الخاص بالحرف S ، ونبحث عن الحرف الأول من الشفرة وهو O ، وعندما نجده نأخذ الحرف الأول في نفس العامود ، وهنا في هذه الحالة الحرف الأصلي هو H . وهكذا مع باقي الحروف .

شفره فجينير تلقائية المفتاح *THE AUTO-KEY VIGENERE CIPHER*

شفرات فجينير بأنواعها المختلفة متشابهة فيما بينها بشكل كبير ، لكن يوجد فرق واضح بين كل نوع أخر ، في شفره فجينير تلقائية المفتاح ، التشفير يكون بنفس الطريقه ، وجمله التشفير أيضا ، لكن في حال انتهت جملة التشفير وحتى نتجنب تكرر المفتاح بعد انتهائه نقوم بوضع النص الأصلي ، أي أن النص الأصلي يدخل في عمليه التشفير .

نأخذ مثال يوضح هذا النوع من الشفرات ، أريد أن أشفر النص التالي:
LIGHT SPEED CHEWIE NOW

بالمفتاح (جملة التشفير ، المفاتيح ، كلها نفس المعنى) التالي : ARGH

أول خطوه نقوم بها هي تشفير الحرف الأول في النص الأصلي مع الحرف الأول في جمله التشفير
L + A = L
I + R = Z
G + G = M
H + H = O
إلى هنا نلاحظ انتهاء المفاتيح (وبدل من إعادته مره ثانيه كما حدث في الطريقة السابقة) نقوم

بإدخال النص الأصلي كمفتاح
الحرف الأول من النص الأصلي L
L + T =E
الحرف الثاني من النص الأصلي I
I + S = A
و هكذا إلى أن ننتهي من تشفير النص.

ويفضل دائما كتابة النص ، والمفتاح ، والنص المشفر ، **بهذه الطريقه للتسهيل** :

Plaintext	L	I	G	H	T	S	P	E	E	D	C	H	E	W	I	E	N	O	W
Key	A	R	G	H	L	I	G	H	T	S	P	E	E	D	C	H	E	W	I
Ciphertext	L	Z	M	O	E	A	V	L	X	V	R	L	I	Z	K	L	R	K	E

ولفك التشفير ، نقوم بفك الحرف الأول من النص المشفر مع المفتاح الأول ، والحرف الثاني من النص المشفر مع المفتاح الثاني ... ونستمر هكذا إلى أن تنتهي المفاتيح.

ولكي نقوم بفك الحرف المشفر التالي (بعد انتهاء المفاتيح) ، نقوم بفك تشفير الحرف المشفر مع المفتاح (الذي يكون عبارة عن أول حرف أصلي حصلنا عليه بعد أول عمليه فك تشفير) . أي أن المفتاح (بعد انتهاء المفاتيح) يكون هو الحرف الأصلي الذي حصنا عليه ، وبعدها يكون الحرف الأصلي الثاني الذي حصنا عليه . (أي أن النص الأصلي يكون هو المفتاح) .

في هذا النوع من التشفير يجب أن تكون العملية دقيقه ، لأن أي خطأ واحد في الحروف قد ينتج عنها خطأ في جميع الحروف التالية ، أيضا عند إرسال رسائل من هذا النوع يجب التأكد من عدم وجود أخطاء في الإرسال.

شفره فجينير طويلة المفتاح *THE Running KEY VIGENERE CIPHER*

(الترجمة ليست ترجمة حرفية صحيحة ، ولكنها صحيحة تماما في المعنى)

هنا في هذا النوع من الشفرات ، يجب اختيار جمله تشفير(مفاتيح) بحيث تكون أطول من النص الأصلي (لاحظ اسم الشفرة) ، وأخذ هذه الجملة ممكن يكون من كتاب ما أو مجله أو أي نص طويل ، يجب أيضا أن تكون موجودة عند الطرف الأخر ، أو اختيار أي جمله للتشفير وإرسالها إلى الطرف الأخر (المهم أن تكون طويلة).

مثال ، أريد تشفير النص التالي: TORA TORA TORA

بجملة التشفير التالية :
AND GOD SAID LET THERE BE LIGHT

لاحظ الجملة أطول من المفتاح ، وطريقه التشفير تكون عاديه كما كنا نشفر من قبل.

النتيجة بعد التشفير:

Plaintext	T	O	R	A	T	O	R	A	T	O	R	A
Key	A	N	D	G	O	D	S	A	I	D	L	E
Ciphertext	T	B	U	G	H	R	J	A	B	R	D	E

وفك التشفير هو العملية العكسية ، وبنفس الأسلوب .

كسر شفرات *AUTO-KEY AND RUNNING KEY VIGENERE CIPHERS*

بالرغم من أنه هذه الطرق أقوى من الطرق السابقة Monoalphabetic، حيث لا تقوم هذه الطرق بتكرار المفتاح كما هو الحال مع الطرق السابقة ، الإ أنها معرضه للخطر أيضا من طريقه التحليل الإحصائي ، وذلك نظرا لأن النص الأصلي يتكرر بدل من تكرار المفاتيح(في حالة الشفرة تلقائية المفتاح) ، وهذا الأمر يشكل خطرا كبيرا.

شفرات *PolyGram Substitution Cipher*

(في بعض الكتب ، يطلق عليها بالـ Polygraphic)

لاحظنا في الطرق السابقة أن أخذ حرف واحد وتشفيره بمفتاح إلى حرف مشفر ، هي طرق ضعيفة ويمكن كسرها بسهوله ، لكن هنا في الـ **POLYGRAM** التشفير سوف يكون بطول بلوك Block ، أي نأخذ البلوك الأول كاملا ونشفره ، ونضع البلوك المشفر . طبعا لا يشترط أن يكون بلوك النص الأصلي هو نفس حجم بلوك النص المشفر.

مثلا لدى خوارزمية تأخذ بلوك من 8 أحرف ، وتضع بدله بلوك مشفر من 8 أحرف ، كما هو موضح بالصورة التالية:

```
AAAAAAAA      maps to      ZXCIJCDV
AAAAAAAB      maps to      APQODFIM
. . .         . . .        . . .
ZZZZZZZZ      maps to      SSTFQQWR
```

اذا أردنا أن نطبق طريقه الـ **Brute-Force** ، سوف نحتاج إلى 26 أس 8 احتمال (26^8)
وهو ما يساوي $26^8 = 208,827,064,576$

وهو طبعا أمر يأخذ زمنا طويلا وحجم كبير جدا جدا في الذاكرة ، باختصار حل غير عملي.

ويتضح هنا أن هذه الشفرات أصعب في الفك من الأنواع السابقة بكثير ، وخاصة في حال كان حجم البلوك كبير بما فيه الكفاية ، وأغلب الخوارزميات الحديثة تأخذ حجم بلوك على الأقل 8 حرف. ولكن في حاله كان حجم البلوك صغيرا ، يمكن كسر هذا النوع باستخدام التحليل المتكرر أيضا.

ومن أشهر شفرات هذا النوع Polygraphic

شفره بلافير Playfair

شفره هيل Hill Cipher

أسطوانة جيفيرسون Jifferson Cylinder ، وبعض الأحيان تسمى **THE BAZERIES CYLINDER .**

شفره بلافير *THE PLAYFAIR CIPHER*

هذه الشفرة تأخذ بلوك Block مكون من حرفين ، والشفرة الناتجة تكون أيضا من حرفين ، وطريقتها تكون بعمل مصفوفة من 25 خانه (5*5) ، نضع في كل خانه حرف أبجدي A و B وهكذا ، وبما أن عدد الحروف الأبجدية (في اللغة الإنجليزية) يساوي 26 ، اذا هناك حرف ليس له مكان ، لذلك هذه الشفرة تضع الحرفين I,J مع بعض في خانه واحده دائما.

المصفوفة ذات بعدين ، 5*5 ، **والصورة التالية توضح شكل المصفوفة:**

A	B	C	D	E
F	G	H	I/J	K
L	M	N	O	P
Q	R	S	T	U
V	W	X	Y	Z

وعاده يكتب في هذه المصفوفة الحروف التي تمثل جملة التشفير (مفاتيح التشفير) ، مثلا لدي جمله التشفير التالية:

The quick brown fox jumped over the lazy dogs

أقوم بعمل المصفوفة 5*5 وأقوم بتعبئة الخانة الأولى بالحرف الأول من جملة التشفير T والخانة الثانية بالحرف الثاني نم جملة التشفير H وهكذا ، (ويشترط) عدم تكرار الحرف الذي ظهر ، أيضا في حال انتهت جمله التشفير نكمل الخانات الباقية بباقي الحروف غير موجودة في المصفوفة.

T	H	E	Q	U
I/J	C	K	B	R
O	W	N	F	X
M	P	D	V	L
A	Z	Y	G	S

أيضا **شكل المصفوفة التالية** ، يكون بعد تعبئتها بجمله التشفير:
Since by man came death

S	I/J	N	C	E
B	Y	M	A	D
T	H	F	G	K
L	O	P	Q	R
U	V	W	X	Z

طريقة التشفير ، كالتالي :

أقوم أولا بتقسيم النص الأصلي إلى مجموعه من البلوكات Blocks كل بلوك من حرفين . لنطلق على الحرفين A,B

قبل أن نبدأ في النظر إلى المصفوفة وبدء التشفير ، ننظر إلى النص الأصلي وبالتحديد إلى كل بلوك مكون من حرفين ، ونرى هل الحرفين متشابهان ، اذا كان كذلك نفصل بينهما بالحرف X .
أيضا في حال كان نهاية النص الأصلي بلوك من حرف واحد ، نضيف له الحرف X .

الآن لبدء التشفير ننظر إلى الجدول:
في حال كان A و B كل منهما في عمود مختلف ، نأخذ المربع الذي يمثل تقاطعهما (الحرفين الذي يمثلان نقطه تقاطع الصف مع العمود).
في حال كان A و B في نفس العمود ، تشفير A هو الحرف أسفله ، تشفير B هو الحرف الذي يكون أسفله (ممكن عمل دوره أي البدء من بداية العمود في حال كان الحرف هو الأخير Wrap .
في حال كان A و B في نفس الصف ، تشفير A هو الحرف على يمينه وتشفير B هو الحرف الذي على يمينه . يمكن عمل دوره Wrapping (اذا لزم الأمر).

مثال ، لدينا المصفوفة التالية (معبئة بجملة التشفير):

L	O	V	E	I/J
S	A	M	N	Y
P	D	R	T	H
G	B	C	F	K
Q	U	W	X	Z

قم بتشفير النص التالي باستخدامها:
AMBASSADOR SHOT

الحل :

الآن كما ذكرنا أول خطوه هي تقسيم النص إلى بلوك يتكون من حرفين.
AM BA SS AD OR SH OT

الخطوة الثانية ، ملاحظه هل هناك بلوك من حرفين متشابهين ، هنا نلاحظ أن هناك بلوك فيه حرفان متشابهان لذلك نضيف X بينهما وهو البلوك الثالث SS ، ليصبح لدينا بعد إضافة X :
AM BA SX SA DO RS HO T

لاحظ أخر حرف وحيد ، لذلك نضيف الحرف X في الأخر ، ليصبح النص الأصلي:
AM BA SX SA DO RS HO TX

نأخذ البلوك الأول ، ونبدأ في النظر إلى الجدول ، الحرف A في نفس صف الحرف M ، اذا ننتقل خطوه لليمين لكل حرف في البلوك الأول .
اذا تشفير A هو M ، وتشفير M هو N .

نأخذ البلوك الثاني ، وننظر في الجدول ، الحرف B في نفس عامود الحرف A ، اذا ننتقل خطوه للأسفل لكل حرف في البلوك الثاني .
اذا تشفير B هو U ، وتشفير A هو D .

نأخذ البلوك الثالث ، وننظر إلى الجدول ، S و X مختلفان في المواقع ،

تشفير S هو الحرف الذي يكون في نفس عمود S وصف X وهو Q
تشفير X هو الحرف الذي سيكون في نفس عمود X وصف X وهو N .

وباقي البلوك تتشفر بنفس الطريقه.

النتيجة النهائية هي:
MN UD QN AM BA MP ID FE

ولفك التشفير ، العملية العكسية ، ولكن في التشفير يجب الذهاب إلى الخانة السفلي في حالة كان الحرفين في نفس العامود ويجب الذهاب إلى الخانة اليمنى في حال كان الحرفين في نفس الصف ، هنا لفك التشفير يحدث العكس ،
تصبح الذهاب إلى الخانة الأعلى في حال كان الحرفين في نفس العامود ، والذهاب إلى الخانة اليسرى في حالة الحرفين في نفس الصف .

نجرب فك MN في نفس الجدول السابق :
M ترجع خطوه للوراء وتصبح A
N ترجع خطوه للوراء وتصبح M .

شفره هيل *Hill Cipher*

تعتبر شفره هيل هي أول شفره تتعامل فيها مع 3 حروف في نفس الوقت ، وسميت بهذا الاسم نسبه إلى مخترعها **Lester S Hill** ، وهي تعتمد في عملها على الجبر الخطي . ولكي تستطيع

التشفير بها يجب أن يكون لديك أساسيات التعامل مع المصفوفات (ضرب المصفوفات بالذات) .

قبل أن نبدأ بالتشفير ، يجب أن يكون جدول الحرف قريب لديك.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

علينا أولا اختيار المفتاح ، مثلا كان مكون من تسعه حروف ، سوف تكون المصفوفة (الخاصة بالمفاتيح) 3*3 ، أي ثلاثة صفوفه وثلاثة أعمده.

مثلا ، لدي جمله التشفير التالية : GYBNQKURP
بعد أعطاء كل حرف قيمته ، نقوم بوضعه داخل المصفوفة على شكل 3*3 ، وتكون شكل المصفوفة كالتالي :

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix}$$

وليكن النص الأصلي هو: ACT ، وفي حال كان أكبر من ذلك يتم تقسيمه إلى بلوكات ، كل واحد يتكون من ثلاثة حروف .

نقوم بوضع النص الأصلي داخل مصفوفة 1*3:

$$\begin{pmatrix} 0 \\ 2 \\ 19 \end{pmatrix}$$

الآن نقوم بعمليه ضرب المصفوفتين ، نضرب الصف الأول في المصفوفة الأولى بالعمود في المصفوفة الثانية نضع الناتج في المصفوفة الجديدة . وهكذا لباقي الصفوف نقوم بضربها بالعمود. ونأخذ الناتج بعمليه باقي القسمة MOD 26 .

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \\ 19 \end{pmatrix} \equiv \begin{pmatrix} 67 \\ 222 \\ 319 \end{pmatrix} \equiv \begin{pmatrix} 15 \\ 14 \\ 7 \end{pmatrix} \mod 26$$

اذا الناتج من هذا النص بعد تحويل هذه الأرقام إلى حروف (بمساعده جدول الحروف) ، أي النص المشفر هو: POH

ولفك التشفير ، كل ما عليك هو إيجاد معكوس المصفوفة ، وتقوم بضربه في النص المشفر مع أخذ باقي القسمة على 26 ، كما هو موضح بالصورة:

$$\begin{pmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{pmatrix} \begin{pmatrix} 15 \\ 14 \\ 7 \end{pmatrix} \equiv \begin{pmatrix} 260 \\ 574 \\ 539 \end{pmatrix} \equiv \begin{pmatrix} 0 \\ 2 \\ 19 \end{pmatrix} \mod 26$$

هناك في الحقيقة أنواع كثيرة من هذه الشفرات ، منها **3-Hill Cipher** ، وفي هذه الحالة المصفوفة يجب أن تكون من 3*3 وهو الذي تحدثنا عنه من قليل، وفي حال النوع **2-Hill Cipher** يجب أن تكون من 2*2 ، وبشكل عام اذا كان لدينا شفره من **n-Hill Cipher** فأنه سوف يكون لدينا مصفوفة من n*n .

مثال على **2-Hill Cipher** ، و هنا سوف لدينا مصفوفة مكونه من 2*2 ، تحتوي على حروف اللغة 26 حرف .
ولكي نستخرج "معكوس المصفوفة" الصحيح عند الفك ، يجب أن تكون **Determinant** هذه المصفوفة أولي مع العدد 26 ، أي أن القاسم المشترك الأكبر ل **Determinant** و 26 يساوي 1 .

الصوره التالية توضح كيفيه التشفير وفك التشفير :

- key: a 2×2 matrix of elements from $\mathbb{Z}_{26}$ $\ni$

يجب أن يكون القاسم المشترك الأكبر ل det و26 يساوي 1

$\det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = (ad - bc)$ is relatively prime to 26.

- Encryption:

ويكون التشفير عن طريق ضرب المصفوفه مع أول حرفين من النص الاصلي مع أخذ باقي القسمه على 26 على الناتج

$$\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} \pmod{26}.$$

That is:

$$c_1 = (a \cdot p_1 + b \cdot p_2) \bmod 26$$
$$c_2 = (c \cdot p_1 + d \cdot p_2) \bmod 26$$

Good diffusion
Poor confusion

- Decryption:

فك التشفير يتم عن ضرب معكوس المصفوفه مع أول حرفين من النص المشفر وأخذ الناتج في عمليه باقي القسمه على 26

$$\begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \pmod{26}.$$

الآن السؤال هو كيف يتم إيجاد معكوس المصفوفة ؟

هناك الكثير من الطرق لإيجاد المعكوس ، منها البسيط ، ومنها المتوسط ، وسوف نستعرض أسهل طريقة لإيجاد معكوس مصفوفة من نوع 2*2 . **الصوره التالية توضيح كيفيه إيجاده :**

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = d^{-1}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

Determinant of a matrix A, denoted by det A :

-- if A(aij) is 2×2, then det A =a11a22 – a12a21

-- if A(aij) is 3×3, then det A =

a11a22a33 + a12a23a31 + a13a21a32 - a13a22a31 - a12a21a33 - a11a23a32

الآن لإيجاد d^-1 (معكوس d) يجب أن يتوفر لدينا معكوس ما بحيث حاصل ضرب هذا المعكوس مع ال d مع أخذ الناتج 26 mod ليكون الباقي 1 .

d * d` = 1 (mod 26)

وهذه الصوره تبين معكوس الأعداد "الأولية" من 2 إلى 26 :

a	1	3	5	7	9	11	15	17	19	21	23	25
a^{-1}	1	9	21	15	3	19	7	23	11	5	17	25

مثال للتشفير وفك التشفير :

$$A = \begin{pmatrix} 9 & 4 \\ 5 & 7 \end{pmatrix}$$

تم أختيار المصفوفه ،والتأكد من أن لها معكوس

$$\det A = 9 \cdot 7 - 4 \cdot 5 = 43 \equiv 17 \pmod{26}$$

$$17^{-1} \bmod 26 = 23$$

تم إيجاد المعكوس

$$A^{-1} = 23\begin{pmatrix} 7 & -4 \\ -5 & 9 \end{pmatrix} \equiv \begin{pmatrix} 5 & 12 \\ 15 & 25 \end{pmatrix} \bmod 26$$

"fo" $\longrightarrow$ 5 14

هنا تم تشفير الحرفين fo والناتج هو xt

$$\begin{pmatrix} 9 & 4 \\ 5 & 7 \end{pmatrix}\begin{pmatrix} 5 \\ 14 \end{pmatrix} \equiv \begin{pmatrix} 23 \\ 19 \end{pmatrix}$$

23 19 $\longrightarrow$ "xt"

قم بضرب الناتج مع معكوس المصفوفه وسوف يخرج الناتج الأصلي

نأخذ مثال شامل للتشفير وفك التشفير بهذه الطريقه :

لدينا النص التالي : THE PROFESSOR IS EVIL
ونريد تشفيره من البروفيسور ☺ .

الآن أولا نختار النوع الذي نريد نم شفره hill هل هو 2 ، أو 3 أو 4 .

ونختار 2-Hill Cipher ، لأنه النوع الأسهل .

الآن نقوم في البداية بتقسيم النص الأصلي إلى مجموعات Block كل واحد منها مكون من حرفين (لأننا اخترنا 2-Hill Cipher، في حاله اخترنا n سوف نقسم الحروف في النص الأصلي إلى n حرف) .

(لاحظ أننا سنعتمد هنا أن الحرف الأول A تكون قيمته 1 وليس 0 ، كما كنا نشفر في الشفرات السابقة ، طبعا يمكنك أتباع الأسلوب القديم وهو الحرف A تكون قيمته 0) .

T H	E P	R O	F E	S S	O R	I S	E V	I L
20 8	5 16	18 15	6 5	19 19	15 18	9 19	5 22	9 12

ونقوم الآن بوضع هذه الأرقام في مصفوفة النص الأصلي P .

$$P = \begin{pmatrix} 20 & 5 & 18 & 6 & 19 & 15 & 9 & 5 & 9 \\ 8 & 16 & 15 & 5 & 19 & 18 & 19 & 22 & 12 \end{pmatrix}$$

الآن تختار مصفوفة التشفير والتي يجب أن يكون لها معكوس حتى نستطيع فك التشفير .

$$A = \begin{pmatrix} 5 & 6 \\ 2 & 3 \end{pmatrix}$$

(يمكنك التأكد من ذلك ، وذلك عن طريق أخذ det وضربه في عدد ما (المعكوس) واخذ الناتج 26 mod ويجب أن يكون الباقي يساوي1 ، وهنا نتأكد من أن المصفوفة صحيحة ولها معكوس) .

الآن نقوم بضرب المصفوفة p في المصفوفة A لكي ينتج لدينا المصفوفة c (الشفره)

$$E = AP = \begin{pmatrix} 148 & 121 & 180 & 60 & 209 & 183 & 159 & 157 & 117 \\ 64 & 58 & 81 & 27 & 95 & 84 & 75 & 76 & 54 \end{pmatrix}$$

الآن نأخذ الناتج 26 mod (لجميع القيم في المصفوفة بالطبع) . ويخرج لدينا النص المشفر :

$$E = AP = \begin{pmatrix} 18 & 17 & 24 & 8 & 1 & 1 & 3 & 1 & 13 \\ 12 & 6 & 3 & 1 & 17 & 6 & 23 & 24 & 2 \end{pmatrix}$$

الآن نقوم بتحويل هذه القيم إلى أعداد .

$$E = AP = \begin{pmatrix} R & Q & X & H & A & A & C & A & M \\ L & F & C & A & Q & F & W & X & B \end{pmatrix}$$

ونقوم بترتيبها بعد ذلك ، ويخرج لدينا النص المشفر التالي :
RLQFXCHAAQAFCWAXMB

الآن لفك التشفير ، نقوم بتجميع الحروف المشفرة في بلوك مكون من حرفين (كما في التشفير) ومن ثم نقوم بضرب المصفوفة في معكوس المصفوفة ، ويخرج لدينا النص الأصلي (طبعا تذكر أخذ باقي القسمة على 26 في حاله كان العدد الناتج أكبر من 26 . لماذا نقوم بذلك دائما ؟ الجواب ببساطه لأننا نريد حصر الناتج في نطاق الحروف الأبجدية ، فإذا كانت الأبجدية تتكون من N حرف اذا سوف نأخذ باقي القسمة على n وهو ما يعرف بـ **modular arithmetic**).

الآن بضرب المعكوس في النص المشفر ، وأخذ الناتج 26 mod ، يخرج لدينا :

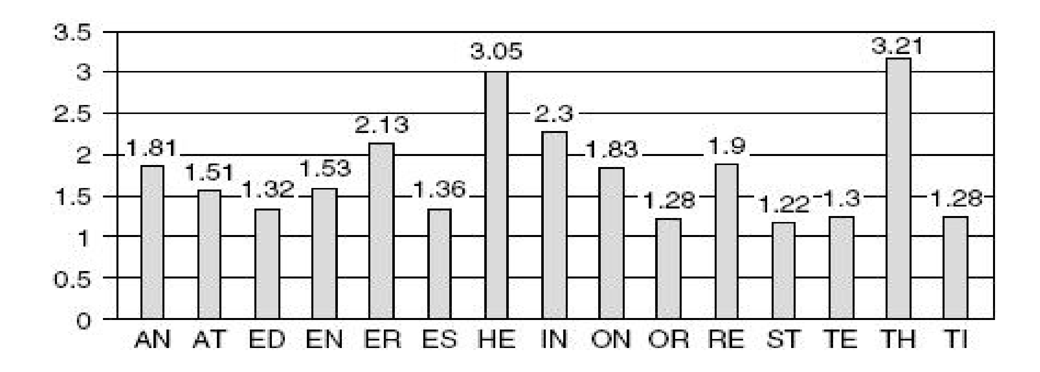
$$D = A^{-1} E = \begin{pmatrix} 20 & 5 & 18 & 6 & 19 & 15 & 9 & 5 & 9 \\ 8 & 16 & 15 & 5 & 19 & 18 & 19 & 22 & 12 \end{pmatrix}$$

ونقوم بتحويله إلى أحرف : ليخرج لدينا :
THE PROFESSOR IS EVIL
و هو النص الأصلي

على العموم طريقه التشفير هذه ضعيفة أيضا وخاصة لطريقه **row reduction** حيث تستطيع كسر هذه الشفره ببساطة ، وسوف نتطرق لها في النسخة القادمة بإذن الله) .

كسر الخوارزميات من نوع PolyGram Substitution Cipher

بما أن الطريقتان التي تطرقنا لهم قبل قليل ، يكون حجم البلوك فيها صغير (من حرفين في حاله شفره بلافير) ، فإنه يمكن كسر هذه الشفرات أيضا ، وخاصة في حال توفر نص مشفر بما فيه الكفاية ، لإجراء التحليل ، لكن هنا سوف يكون البحث عن أكثر حرفين تكرروا في الشفرة ، واحتمال أن يكونوا TH أو HE ، الجدول التالي يبين أكثر حرفين Digraphs يتكرروا في اللغة



وفي حاله كان حجم البلوك من 3 أحرف trigraphs ، هناك أيضا كلمات تتكون من ثلاثة أحرف تتكرر أكثر من غيرها مثل THA و THE و AND وغيرها .و في الشفرات الحديثة Modern Cryptography تم القضاء على ضعف هذه الطرق ، حيث يكون حجم البلوك على الأقل 8 أحرف .

أسطوانة جيفيرسون THE JEFFERSON CYLINDER

اسطوانة جيفيرسون واحده من أقوى الأجهزة التي استخدمت في التشفير، حيث الشفرة الناتجة قويه ولا يمكن كسرها بسهوله أبدا ، إلا في حاله سرقة الجهاز بأكمله ، الاسم جيفيرسون يعود إلى اسم مخترعها **توماس Thomas Jefferson** .

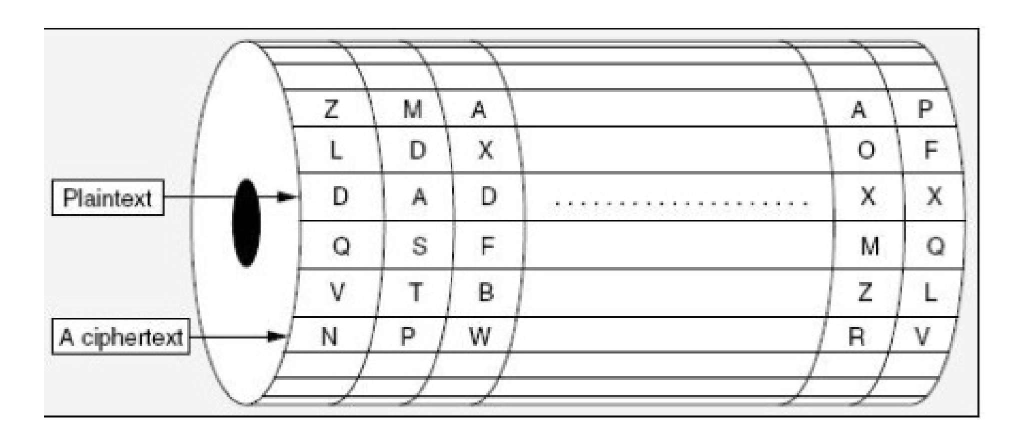
هذه الإسطوانه تتكون من 36 عجله بجانب بعض ، و26 صف ، أي أن كل صف به 36 عجله (حرف) .

في حال أردت التشفير النص الأصلي ، يجب أن أضع جميع العجلات في صف ما في شكل النص الأصلي ،أي أقوم بتحريك العجلة الأولى مثلا في الصف الرابع إلى الحرف المراد ، الآن أحرك العجلة الثانية في نفس الصف إلى الحرف الثاني المراد تشفيره ، ونفس الكلام لباقي الحروف لكن في نفس الصف.

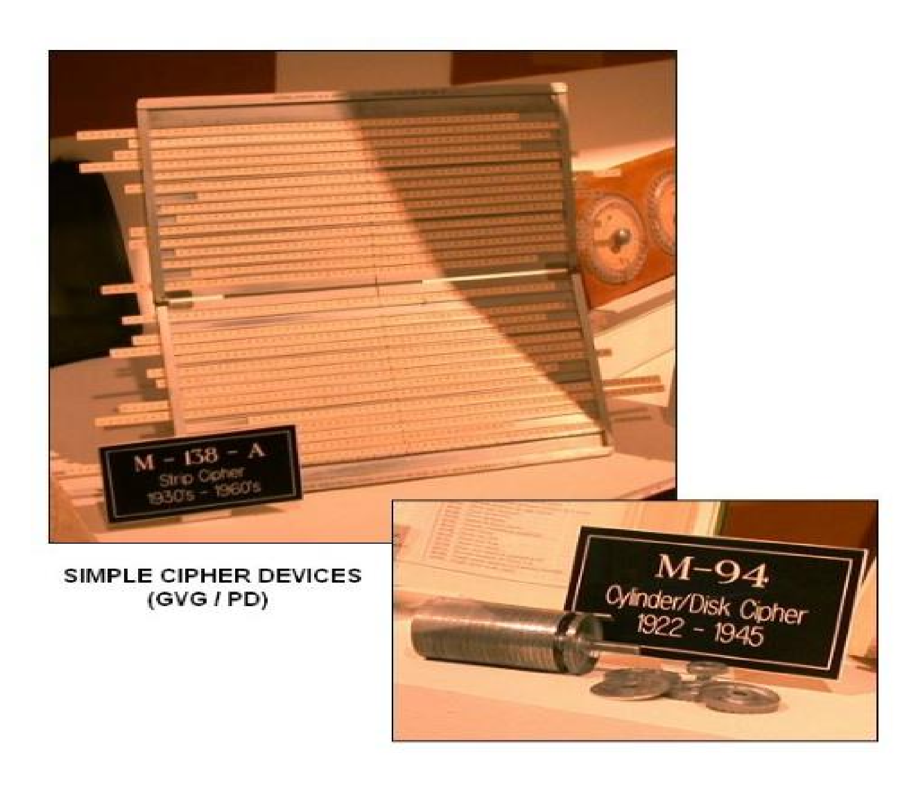
الآن بعد وضع الصف كامل على النص الأصلي ، أقوم باختيار أحد الـ 25 صف المتبقية ، أي هناك 25 شفره ممكنه، وأرسل هذا النص للطرف الأخر.

في حاله فك التشفير ، يقوم بترتيب النص المشفر في صف ، بعدها ينظر إلى باقي الـ 25 صف ويشاهد من هو النص الأصلي ، أي يقوم بالبحث في جميع هذه الصفوف ، حتى يستطيع الحصول على النص الأصلي .

صوره لتوضيح أسطوانة جيفيرسون:



وهذه صوره لها في الواقع :

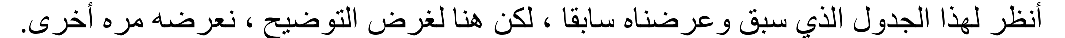


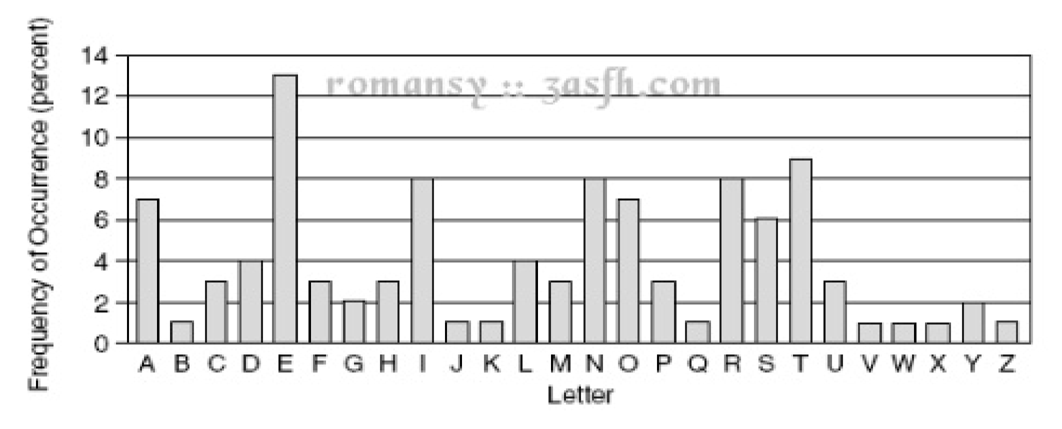
SIMPLE CIPHER DEVICES
(GVG / PD)

وبالرغم من قوه هذه الطريقه في التشفير ، فإنها لم تنتشر نظرا لصعوبة تطبيقها **Hard to Implementation** ويجب على الطرفين الاحتفاظ بهذه الإسطوانه ، وفي حال وقعت في أيدي العدو ، فيمكن كسر التشفير بتلك الطريقه بكل سهوله.

التشفير بطريقه الـ *HOMOPHONIC SUBSTITUTION CIPHERS*

طريقه الـ **HOMOPHONIC** تعتبر من الطرق الجيدة لإحباط التحليل الإحصائي ، الذي لن يستطيع فعل شيء لمثل هذا النوع من الشفرات ، حيث كما نعرف أنه هناك حروف تتكرر أكثر من غيرها في اللغة ، في هذه الطريقه كل حرف من هذه الحروف التي تتكرر كثيرا ، يكون لها أكثر من احتمال ، أي تتشفر إلى أكثر من حرف وبطريقه عشوائية.


أنظر لهذا الجدول الذي سبق وعرضناه سابقا ، لكن هنا لغرض التوضيح ، نعرضه مره أخرى.



لاحظ أن أكثر حرف يتكرر هو E بنسبه 13 و T بنسبه 9 ، المهم هو أننا نقوم بعمل جدول يسمى جدول الـ Homophonic يحتوي على تشفير الحرف E ب 13 طريقه ، و Tب9 مرات ، أيضا من المهم ذكر أن هذا النوع من الشفرات يشفر الحرف الواحد في النص الأصلي إلى حرفين وهو ما يعرف بـ one–to–many mapping .

أنظر إلى جدول الـ Homophonic :

Plaintext letter	Choices for ciphertext unit												
A	BU	CP	AV	AH	BT	BS	CQ						
B	AT												
C	DL	BK	AU										
D	BV	DY	DM	AI									
E	DK	CO	AW	BL	AA	CR	BM	CS	AF	AG	BO	BN	BE
F	BW	CM	CN										
G	DN	BJ											
H	AS	CL	CK										
I	DJ	BI	AX	CJ	AB	BP	CU	CT					
J	BX												
K	DI												
L	AR	BH	CI	AJ									
M	DH	BG	AY										
N	BY	DG	DF	CH	AC	BR	DU	DT					
O	DZ	BF	DX	AK	CG	BQ	DR						
P	BZ	DE	AZ										
Q	DD												
R	AQ	DC	DQ	AL	CE	CF	CV	DS					
S	AP	AN	AO	CD	DW	DV							
T	CB	DB	DP	CC	AD	CY	CW	CX	AE				
U	CA	AM	BA										
V	BB												
W	CZ												
X	BD												
Y	DO	DA											
Z	BC												

مثلا ، أريد تشفير العبارة:
RETREAT

نبدأ بالحرف الأول R ، له أكثر من 8 شفرات في الجدول ، ويمكنك اختيار ما تشاء.
الحرف الثاني E ، له 13 شفره ، ويمكنك اختيار ما تشاء.

الآن وبعد أن قمت بالتشفير ، الناتج هو:
DQ AW CC AQ CO BS DB

(الناتج قد يختلف عن الناتج الخاص بك ، لأنني قد اخترت شفرات لم تختارها) .

ولفك التشفير ، يمكنك بناء **جدول عكسي inverse mappings** للجدول السابق ، **كما في الصوره التالية:**

Ciphertext Pair	Letter	Ciphertext Pair	Letter	Ciphertext Pair	Letter	Ciphertext Pair	Letter
AA	E	BA	U	CA	U	DA	Y
AB	I	BB	V	CB	T	DB	T
AC	N	BC	Z	CC	T	DC	R
AD	T	BD	X	CD	S	DD	Q
AE	T	BE	E	CE	R	DE	P
AF	E	BF	O	CF	R	DF	N
AG	E	BG	M	CG	O	DG	N
AH	A	BH	L	CH	N	DH	M
AI	D	BI	I	CI	L	DI	K
AJ	L	BJ	G	CJ	I	DJ	I
AK	O	BK	C	CK	H	DK	E
AL	R	BL	E	CL	H	DL	C
AM	U	BM	E	CM	F	DM	D
AN	S	BN	E	CN	F	DN	G
AO	S	BO	E	CO	E	DO	Y
AP	S	BP	I	CP	A	DP	T
AQ	R	BQ	O	CQ	A	DQ	R
AR	L	BR	N	CR	E	DR	O
AS	H	BS	A	CS	E	DS	R
AT	B	BT	A	CT	I	DT	N
AU	C	BU	A	CU	I	DU	N
AV	A	BV	D	CV	R	DV	S
AW	E	BW	F	CW	T	DW	S
AX	I	BX	J	CX	T	DX	O
AY	M	BY	N	CY	T	DY	D
AZ	P	BZ	P	CZ	W	DZ	O

و هو نفسه الجدول السابق ، لكن تم وضع أي شفره في صف لكل شفره ، وما يقابلها من النص الأصلي . ويصبح فك التشفير أمر في غاية السهولة!

وبالرغم من قوه هذه الطريقه وشفراتها الأمنه ، الإ إنها لم تستخدم بشكل كبير ، لأنها كما لاحظنا تعتمد على اللغة ، والحروف التي تتكرر فيها كثيرا ، على عكس الشفرات الحديثة التي لا تعتمد إطلاقا على اللغة

النوع الثاني من أنواع التشفير بالطرق الكلاسيكية هو :

التشفير بالإبدال *TRANSPOSITION CIPHERS*

وطريقه الإبدال لها العديد من الأشكال والطرق والشفرات ومنها البسيط ومنها المعقد ، وسنأخذ أشهر وأسهل اثنين:

الطريقه الأولى طريقه العكس Reversing :
الطريقه بسيطة للغاية ، وكل ما في الأمر أني سأبدل الحرف الأول مكان الحرف الأخير ، الحرف الثاني بالحرف ما قبل الأخير ، وهكذا ، وهي من أضعف أنواع الشفرات ، هذا اذا اعتبرت شفره من الأساس!

مثال:
النص الأصلي : Wajdy Essam Is Java Developer
بعد تشفيره : repoleveD vavJ sI massE ydjaW

وبعد تقسيمها إلى بلوكات (من 5 حروف) ، يصبح لدي :
repol eveDv avJsI massE ydjaW

طريقه أخرى للعكس ، ولكن بشكل منظم :

مثلا نقوم بتغير أماكن الحروف في كل بلوك بطريقه معينه ، مثلا نجعل:
الحرف 1 مكان الحرف 4 ، الحرف 2 مكان الحرف 3 ، الحرف 3 مكان الحرف1 ، الحرف 4 مكان الحرف 5 . الحرف 5 مكان الحرف 2

ونقوم بترتيب التغييرات السابقة ، لنسهل من العملية :
الحرف 1 مكان الحرف 4
الحرف 4 مكان الحرف 5
الحرف 5 مكان الحرف 2
الحرف 2 مكان الحرف 3
الحرف 3 مكان الحرف 1
الحرف 1 مكان الحرف 4
(رتبنا هذا التبديل ، عن طريق جعل الحرف الأخير هو الأول ، هذه العملية للتسهيل فقط).

نفرض أن لدينا النص التالي: THE SKY FALLING PLEASE ADVISE

ونريد تشفيره بهذه الطريقه ، نقوم أولا بفصل كل 5 حروف في بلوك واحد

ليصبح النص الأصلي هو: THESK YFALL INGPL EASEA DVISE

الآن نقوم ببدء التغييرات ، كل بلوك على حده ،

نبدأ بالبلوك الأول :
الحرف 1 مكان الحرف 4 ، نطبقه لينتج : الحرف T مكان الحرف S
الحرف 4 مكان الحرف 5 ، نطبقه لينتج : الحرف S مكان الحرفK

و هكذا لباقي الحروف في البلوك ، ونفس الأمر مع باقي الـ Blocks في النص الأصلي ، بعد أن نقوم بعلميه التشفير ، يخرج لدينا النص المشفر التالي:
EKHTS ALFYL GLNIP SAAEE IEVDS

والتشفير بهذه الطريقه ضعيف للغاية ، إذ يمكن لأي شخص قوي الملاحظة معرفه النص الأصلي بمجرد النظر والملاحظة على البلوك ، لكن في حال استخدمت هذه الطريقه الإبدال مع الطريقه الأخرى الإحلال سوف ينتج شفره قويه جدا ، وأغلب الشفرات الحديثة تعتمد على هذا المفهوم (**الدمج بين طريقه Transposition وطريقه Substitution**) .

الدمج بين طريقتي الإبدال والإحلال *COMBINATION SUBSTITUTION/TRANSPOSITION CIPHERS*

نأخذ مثال بسيط ، لتوضيح كيفيه عمل شفرات من هذا النوع ، والشفرة الذي سنتبعها تتكون من ثلاث خطوات علينا المرور بها

الخطوة الأولى ، ليكن لدينا الجدول التالي ،الذي سنستخدمه في التشفير مرتين . **أنظر صوره الجدول :**

	A	B	C	D	E
A	A	B	C	D	E
B	F	G	H	I	J
C	K	L	M	N	O
D	P	Q/Z	R	S	T
E	U	V	W	X	Y

الآن مثلا أريد أن أشفر الحرف A ، الناتج هو الحرف في أول الصف ، والحرف الذي في أول العمود . أي أن A تتشفر إلى الحرفين AA (أي أخذ العامود والصف الذي يقع فيه الحرف المراد تشفيره).

يفضل تغيير كلمه تتشفر عند استخدام الجداول إلى "اختيار" أو بالإنجليزي Maps .

B maps to AB
C maps to AC
. . .
Q maps to DB
. . .
Y maps to EE
Z maps to DB

مثال ، ليكن لدي العبارة التالية (النص الأصلي) : TAKE ME TO YOUR LEADER

أبدأ الآن بالتشفير ، الحرف T يتشفر maps إلى DE . الحرف الثاني A يتشفر Maps إلى AA . وهكذا ، لينتج لدي الشفرة:
DE AA CA AE CC AE DE CE EE CE EA DC CB AE AA AD AE DC

انتهت الخطوة الأولى ، التي كل فيها تشفير الحرف في النص الأصلي إلى حرفين.

الخطوة الثانية أخذ النص المشفر وأقسهم على قسمين(صفين) ، وأخذ الحرف الأول من الصف الأول و الحرف الأول من الثاني ، وأخذ الحرف الثاني من الصف الأول و أخذ الحرف الثاني من الصف الثاني ، و هكذا....

النص المشفر:
DE AA CA AE CC AE DE CE EE CE EA DC CB AE AA AD AE DC

بعد تقسيمهم إلى صفين ، ينتج لدي:
DE AA CA AE CC AE DE CE EE
CE EA DC CB AE AA AD AE DC

أبدأ بأخذ الحرف D و C ليكون البلوك الأول ، Eو E ليكون البلوك الثاني ، وهكذا ، لينتج لدي:
DC EE AE AA CD AC AC EB CA CE AA EA DA ED CA EE ED EC

الخطوة الثالثة ، أقوم بالرجوع إلى الجدول السابق ، الذي ذكرنا أننا سنستخدمه مرتين ، ونقوم بتشفير كل بلوك:

نأخذ البلوك الأول DC ويتقاطعا في الجدول عند الحرف R .
نأخذ البلوك الثاني EE ويتقاطعا في الجدول عند الحرف Y .
ونستمر هكذا ، لينتج لدي:
RYEANCCVKOAUPXKYXW

الآن أضعهم في شكل Block كل واحد يتكون من 5 حروف:
RYEAN CCVKO AUPXK YXW

وانتهت عمليه التشفير، معقده قليلا ، أليس كذلك ! **لكنها ممتازة جدا.**

ولفك التشفير ، نقوم بالعملية العكسية ، لنفك الشفره الناتجه من عمليه التشفير السابقة:
RYEAN CCVKO AUPXK YXW

نبدأ بأخذ الحرف الأول ونشاهد الجدول ونأخذ الحرفين الذي نقطه تقاطعهما الحرف المراد ، ونضع الحرف الأول في الصف الأول ، و الحرف الثاني نضعه في الصف الثاني ، وهنا أول حرف في الشفره هو R ، ننظر إلى الجدول ، نلاحظ أن الحرفين D و C نقطه تقاطعهما هو R ، لذلك نضع الحرف الأول D في الصف الأول ، والحرف الثاني C في الصف الثاني ، ونستمر بهذه الطريقه ، إلى أن ننتهي . والناتج هو :

DE AA CA AE CC AE DE CE EE
CE EA DC CB AE AA AD AE DC

الآن نبدأ بفك التشفير من الصف الأول ، نأخذ بلوك بلوك من الصف الأول ، وعندما ينتهي نبدأ بالصف الثاني :
DE map to T
AA map to A
CA map to K
و هكذا...

لينتج لدينا النص الأصلي : TAKEMETOYOURLEADER

قم بترتيبه بحيث يصبح مقروء ، لينتج: TAKE ME TO YOUR LEADER

هذه الشفره السابقة ، مشابه لشفره ألمانية استخدمت في الحرب العالمية أسمها **ADFGVX** وعلى الرغم من بساطة الفكرة ،الإ إنها عقدت كاسري الشفره لفترة من الزمن . ولكنها كسرت في النهاية على يد كاسر فرنسي ، لكن بعد أخذ الكثير من الوقت والجهد في المحاولات.

الشفره الأمنه *THE ONE-TIME PAD*

(الكثير من الترجمات تترجم هذه الشفره ب "شفره الوسادة الكاملة" ، وهي ترجمه سيئة للغاية) .

هذه الشفره هي الشفره الأكثر أمانا على مدى تاريخ التشفير ، لم ولن يستطيع أحد كسر شفرات هذا النوع أبدا ، واستخدمت هذه الشفرات في الكثير من الحكومات وأجهزه الاستخبارات كما قرأت سابقا عنها.

طريقه الشفره كالتالي ، هو عمل (مثلا نطلق عليه) كتاب **one-time pad** ، بداخل هذا الكتاب توجد صفحات Sheets بداخل كل صفحه من هذه الصفحات أرقام عشوائية لا تتكرر أبدا ، هذه الأرقام العشوائية تمثل الازاحه المستخدمة (أي كل رقم منها هو مفتاح) .

في حال شفرت نص بهذه الطريقه أقوم بإرسال النص المشفر و رقم الصفحة إلى الطرف الأخر ، وأقوم بقطع الصفحة من الكتاب وأحرقها بالنار ، اذا لزم الأمر ☺ ، أي يتم القضاء عليها .

والطرف الأخر يكون لديه نسخه مماثله من الكتاب one-time pad ، ويقوم بفك التشفير عن طريق رقم الصفحة ، و بعدما يتم فك التشفير والحصول على النص الأصلي ، يتم أيضا قطع الصفحة أيضا

مثال ، لدي الشفره التالية : ENGAGE WARP DRIVE

والصفحة الأولى تتكون من:

9 20 13 0 21 1 13 19 9 5 25 12 25 4 7 25 0 8 8 7 24 2 6 18 16 10 23 5 11 12 13 6 22 22 17 3 8 0 0 19 4 15

أقوم بجمع الحرف الأول E مع الرقم الأول 9 لينتج N ، و أجمع الحرف الثاني N مع الرقم الثاني 20 لينتج H . وهكذا....

اذا تبقت أرقام في الصفحة ، أو لم تبقى هناك أي أرقام ، أقوم بتدمير الصفحة ،

شكل النص بعد التشفير:

Plaintext letter	E	N	G	A	G	E	W	A	R	P	D	R	I	V	E						
Shift value	9	20	13	0	21	1	13	19	9	5	25	12	25	4	7	25	0	8	8	7	24
Ciphertext letter	N	H	T	A	B	F	J	T	A	U	C	D	H	Z	L						

Plaintext letter																					
Shift value	2	6	18	16	10	23	5	11	12	13	6	22	22	17	3	8	0	0	19	4	15
Ciphertext letter	باقي الارقام في الصفحه ، لا تستخدم																				

وهكذا نلاحظ أنه يستحيل كسر الشفره هنا ، لأن المفتاح عشوائي ولن يتكرر أبدا ، وفي هذه الحالة لن يتم كسر الشفره.

هذه الطريقه لن تستخدم هذه الأيام بسبب صعوبة الاحتفاظ بهذا الكتاب ، وصعوبة إرساله إلى الطرف الأخر ، أيضا صعوبة أضافه صفحات جديدة فيه.

ولكن في حاله كنت مرسل كتاب إلى الطرف الأخر من قبل ، يمكنك إرسال شفرات بهذا النوع ، ولن يكشفها أي أحد على الإطلاق ، إلا في حال أنكشف الكتاب (one-time pad) !

أسئلة شامله لكل الموضوع

قم بفك هذه الشفره ، علما بأنها مشفره بخوارزمية قيصر :

MXXFT QQHUX WMDYM QHQDO DQMFQ PNKYQ EUZOQ ARAXP AZMOO AGZFA RYKNQ
SUZZU ZSXQE ESDQQ PTMFD QPMZP USZAD MZOQN ADZAR YKOAZ PGOFE BQQOT
MZPFT AGSTF UZAIO AZRQE EABQZ XKMZP RGXXK

هذه الشفره مشفره بخوارزمية فجينير simple Vigenere cipher ، قم بفكها
(أستخدم طريقه كسيسكي لمعرفه طول المفتاح)

SSQYN ASXES RBFOR SOUYK VTAKO QVKSZ WOQSF VNOBB BRWKB BRCQS
QSOSF WJYSX FHKYS YGODI FSUMD BJJOD FQCWN IBSDO HSPBW XBDIL
MWQGP FZNVD DOSGO NEZSB JJSBQ FSXUW QOIOZ VLBIN TSBTP VBKUV
OXKOJ KDFMZ UCUBB DVITS PKTHC ZPZCB FWZVZ YCLMW HJOSO VBQCE
SGSSO BIWCS FDISC BZOBN DFMZU CUBBD VIORS NJHWY OBSGZ CFUTD
FSOUS BWSFV BUAOO SNOTO ZPSSR FBBCY SGQRP HDKVZ OXEJO XTHCX
FGQYU HVKOR PYPYC PBDDV JSRMS MDDPU FKQVM MSQDB FGGBP GSXLS
BXFHV OMSAO OHOBZ BIWCS FDISC BZOBN JHGKQ DZSDO HSPBG LPGHY
OORNJ GCXXS GVFMF YTWBQ NWQRB SZSND ZONSB DJBUO MZWZU WQMVF
JODFM ZUCUB BDVIH FSOOK WMIAO XOWBQ TAWDI FWMIO FNJBH OSBSD
DFMZU CUBBD VICCG DPBON EWGYO KSCMS MCUOZ VJBUC XWZVJ OAMSM
DDPUF KQVMK ORBOU KCBLG SMVFW DZBRO EWHSP BIZQS FCBRR VFFWF
FFDBF BHSDS VKMZG DFDSX TCBXF OZMSM DDPBC WJQCX OSKIP FYZFF
SXOWO VUFOZ QSKKE SOXEK OCIWB QUCBV BKFOO QSSOH FYEIQ DJCBD
PQFIQ HCQSO DRZKW DIQCN JBUDI SCBZI DZFFG KERZO SWJOS DFOOH
WMFVO VMKOI OSFZF HSBEW GKQDS KSWBQ DFMZU CUBBD VIDVS CUBID
IWZVB DDBPT SCTWC XBZ

هذه الشفره مشفره بخوارزمية Auto-Key Vigenere Cipher قم بمعرفه المفتاح
الأساسي بعدها قم بفك الشفره:

TVWFP VVHZD PZXLX ADBSS SSWBW KAABS DXZFG ANWTZ PWEKV AEOEA
PIOBZ TALSV XUIFW AYEMU MFWAY EMWLT AMMNL HGAHX QILIG PPXFQ
ZMEAD XUXCM RSJHZ XLXCW HKNEH YKZMB OEDXZ FGANW TZPWE MOGWO
EAPKH HRTAL SVXUI FWAYE MUMFW AYEMW LTAMM NLHGA HXQIL IGPPX
FSSSW BWKAA BS

هذه الشفره مشفره بخوارزمية Monoalphabetic Cipher قم بفكها:

ULNEA YTWPX TFNUR WBPHN BPEXE YRKXB PANXE YRKFX HNENW WPETF
NUULN BKRFN YZNKU LNSXW LYSUL NWNPP ETULN GXKTW YSULN PXKYZ
NKULN FXZNW UYIHY ZNKPF FULNN PKULP ETYZN KPFFU LNIKN PURKN
WULPU BYZNP FYEAU LNAKY RETWY AYTIK NPUNT BPEXE LXWYD EXBPA
NXEUL NXBPA NYSAY TLNIK NPUNT LXBBP FNPET SNBPF NLNIK NPUNT
ULNBA YTGFN WWNTU LNBPE TWPXT UYULN BGNSK RXUSR FPETX EIKNP
WNXEE RBGNK SXFFU LNNPK ULPET WRGTR NXUKR FNYZN KULNS XWLYS
ULNWN PPETU LNGXK TWYSU LNPXK PETYZ NKNZN KJFXZ XEAIK NPURK
NULPU BYZNW YEULN AKYRE TULNE AYTWP XTXAX ZNJYR NZNKJ WNNTG
NPKXE AMFPE UYEUL NSPIN YSULN DLYFN NPKUL PETNZ NKJUK NNULP
ULPWS KRXUD XULWN NTXEX UULNJ DXFFG NJYRK WSYKS YYTPE TUYPF

FULNG NPWUW YSULN NPKUL PETPF FULNG XKTWY SULNP XKPET PFFUL
NIKNP URKNW ULPUB YZNYE ULNAK YRETN ZNKJU LXEAU LPULP WULNG
KNPUL YSFXS NXEXU XAXZN NZNKJ AKNNE MFPEU SYKSY YTPET XUDPW
WYAYT WPDPF FULPU LNLPT BPTNP ETXUD PWZNK JAYYT PETUL NKNDP
WNZNE XEAPE TULNK NDPWB YKEXE AULNW XOULT PJ

هذه الشفره مشفره بخوارزمية بلافير Playfair Cipher قم بفكها :

PK QT OX OK KR QK ZX BI OZ BZ ZO EK KQ KP ZO IB ZO KG ZS VL HR
OR HY EK RK RU PH BO OW IH KR YK FW EK OI NR KR YK FW EK AF AX
AT VA KU GX OW YH VM EI FL HT QT XG AB LO LZ RH EK KU AE MF QH
AI EK HY KY QE OW IH KR UG FT ZN AI ZS FC LO TL PH TF BZ LZ RH
EK RQ OR RH OL CI ZS XL OF VD RE IK KR HR QK OD VK RO CI EK RH
RQ LO OD VK KZ LI OL NR RL KI EK HU XZ KE AF XK SI LI OW VC KU
QE FW OR HY EK HU XZ KE MW AZ EK HY FW TB KU GX ZS VL LS DS HY
EK HU XZ KE FL FU CI EK HY FW TB KU GX KR WL SD UH IC XZ KE OW
IH KR HR AF UK PH OZ BZ OW IH KR HR AF AG AT OZ BZ EK RY FT OK
FL FU CI ZS XL OF VD RE IK KR HR QK OW KY MU BO KQ RE QR YK ON
KR AF ER KA NI UK MU WF ER AF ER WM TA RA OR RH OU ZS FV LF RE
KR YK YG OW UK OW XL QE FW OR HY EK RQ OR RH OQ YH HE KR YK YG
OW UK MW AZ EK RQ OR RH SW LZ TY RO CI ZN AF XG OU ZS VL LS DS
HY EK KY KY MU BO KU EX OW IH KR HR AF UK PH OZ BZ OW IH KR HR
AF AG AT OZ BZ EK KU HQ IO XI FL FU CI ZS XL OF VD RE IK KR AF
ER EA CI RH EK KU EX QK MS EK RH HY NI IS QT VU LW RU CI UH HI
EF MK UA CI UM YG RU WF CI ZN AF XG OU ZS VL LS DS HY EK HY EF
MK UA CI SL CI OW IH KR MS EK RH HY AF ER KA KR WL SD UH XL RU
OL CI ZS XL FA EK OR ZN AF XK SI LI EK TQ ZS XL OF VD RE IK KR
SL CI ME LI LQ HP KP RE OR ZI BO KY HY QK FW ZO ZM SL ON OL CI
EH KY KU IO EK HU OW IH RL KN RU WM EA ZN NF EK UK YK XY OZ RO
BD NL HF ZO ZK IN KR FT ML TF UA XB ZO XL OW XY RN LO GX IN KR
SL CI ME LI LQ HY PH PK RO NZ IO VU OW KO QK FW ZO KX KY HY LW
DB AT XY BZ NI EK TY HT ZO XL OW IH HR YK XS RU TF BZ MW OY RN
ZN KL KY HY ZO ZN LW DB AT XY BZ NI EK TY HT ZO XL OW XV LI OL
NR RL NZ RN LO ZS IS FL CI EK RH RQ LO HP TQ ZO MS CI EK RH RQ
LO ZA TW ZO ZK KR EK FT XI FL FU CI ZS XL OF VD RE IK KR HR QK
OD VK KO ZO EK KY KY MU BO TU IR KR XB IE UO QE FW OR RE KR FL
FD TA ZR KR OZ VK RO CI UH KI EK RP UK HM RU XG OX ZB OK AZ FW
BX RU OU BO OW XI FL XU OW XF RU KA OW VD RE IK KR GW HU XZ KE
OX ON KR YK YG OW UK AI EK KU EX NI ZX PH OD VK NI ON KR YK FW
EK OW XY RN LO GX SZ LI RF YH RN SZ HR OR OD VK KO EK KP HR OR
RH OD VK KI NI ZS PH TW EK HY UF OW IH KR UH XG UK HA XZ KE IK
SZ LZ RH ON KR OZ VK EK RU SO AZ LF RE KR OK FW XK LI UK KI EK
RP ZO EK KY KY MU BO TU IR KR XB IE SZ DZ HU XZ KE IO EK KY RU
HE KE SZ LZ RH ON KR FL FU CI EK TQ XZ KE OW IH UO QE FW OR HU
XZ KE MW AZ FL HT QT XG OL CI ZS XL FA EK OR ZN AF XK SI LI OW
IH KR HR AF UK PH OZ BZ OW IH KR HR AF AG AT OZ BZ EK RY UA HE
CV UF OW XV LI OL YX UH KI EK RQ OR RH ER YK RA ZN CD DZ ZO VB
HR OR AE KU OW VD RE NX HF AX UD SW LZ RE KR YK FW EK LW TA XG
OK KR YK YG OW UK AI EK KU EX OU ZS VL HR OR HY EK KP HR OR WH
RU EO HR OU IR KR UK SF OW YH PH EF OD ZD BZ OW FV LZ ZO ZK TD
BZ NF EK CR DN KE KR AF ER HE KY RP OL BD NL HF ZO ZK IN KR FT
TX CI OL CI HP RH FQ ZO PK XC FT FL BD NL HF ZO ZK IZ KO TX CI
OW XV LI OL NR RL NZ RN LO ZS IS XV LI CS KU GX HY EK RP OW XL
OF XC RY EA ZN YA SL CI ZO WH RU UK ZO IB AP QK WF CI DY SD OH
KR AF ER TF ON KR UK LO OW VD RE IK KR NX HF OX QB HR LO TU ON
KR YK FW EK OW IH KR HR AF UK PH OZ BZ OW IH KR HR AF AG AT OZ
BZ EK RY YD EK FL FU CI ZS XL OF VD RE IK KR SL CI ME LI LQ HU
DZ ZO VB HR OR AE KU LW DB AT XY BZ NI EK TY HT ZO XL OD PH OK

```
LN HB HR OR AE KU EK OR ZA PH LS IO ZK KR MK UA CI OW FC DO FL
OZ RF SO RU LC LW DB AT XY BZ NI ZN XB ZO FL CI ZN AF XG OU ZS
FV LF RE KR NF DV OW FZ LS OL BD NL HF ZO ZK IN KR FT TX CI OK
KR OD PH OK LN RS BD NL HF ZO ZK IN KR FT TX CI OL CI LS DS EK
HQ HR OR AE KU EK OR ZA PH LS IO ZK KR MK UA CI LW DB AT XY BZ
NI EK TY HT ZO XL OW XV LI OL NR RL NZ RN LO ZS IS IH KR BZ LI
OL YX UH EO AG SR RP OW ZO UA TF RF PK ZO UA HA XT TQ KU GX OW
VD RE IK KR RW AS TU PH HE KR DY LK AI EK KU RU FS CI EK KQ FT
XL AI EK RU FT LZ RH EK HU DZ KU NI BH LZ RH LS DS EK KY RU HE
RL CI LZ RH LS DS EK HQ HR OR AE KU EK OR ZA PH LS IO ZK KR MK
UA CI OU ZS VL HR OR HY RF OZ CT XO AN OZ RF PK ZO EK TY RF PK
AI ZS VC HQ HR OR HY TD GF RF UH OW XD RP LS RK HQ HR OR HY EK
RP ZS XC UH XG UK IH KR RF CI OL YX NI EK RP QK MW OY RI AS OW
XY QB HR LO TY BI QO GW RH DY SD OH KR YK FW EK OW XL SQ YL TY
EH AS TU PH HE KR DY LK AI EK KU RU FS CI EK KQ FT XL AI EK RU
FT OW IL PH HR PH EF OD ZD BZ WH RU EO HR EK OR ZA PH OU ON KR
MK UA CI EK TQ ZS XL OF XY ZX PH IU YQ PH EF UK YK XC RU TF BZ
VU OW NI ON KR MF QH AI EK RQ TL UH YK FW EK OW YH PH EF EH KY
RE RL EK LO MW OY RN ZN KL KY HY ZO ZN IK KR FQ DO DS QK IU AE
AX AT IA IS FL CI NI LS DS EK KQ RU OK OU IR KR YK FW EK OW FL
SD OH KR NX HF OU IR KR OF WF CI LS DS EK HQ HR OR AE KU EK OR
ZA PH IO EK KP TA OQ YH PH EF EK ZO ZK RL EK LO EK KQ HR OR TL
DA YD TY OZ NZ ZX PH YK PH EF MK KY TQ VU OW RI AT IA IS FL CI
ZN AF XG OU ZS XL FA LS DS EK OR KR RL FV LF RU CI ZN AF LG RH
XP SI LI OW IH KR HR AF UK PH OZ BZ OW IH KR HR AF AG AT OZ BZ
EK KU YI EK FL FI
```

طبعا ، في حال أردت الحل ، عليك بطباعه الشفرات حتى تستطيع عد الحروف والحل بدقه أكبر .

القسم الثالث : التطبيق Implementation

في هذا الجزء ، سوف نتناول تطبيق جميع الشفرات التي تطرقنا لها في القسم الأول من الكتيب ، علما بأن تطبيق هذه الشفرات سوف يكون بلغه سي++ ، ولن نستخدم سوى مفاهيم أساسيه في اللغة وتجاهلنا نقاط قوه اللغة (مثل مفاهيم Exception , Template , oop) فالغرض هنا شرح هذه الشفرات بأسهل أسلوب ممكن . علما بأن البرامج كاملة موجودة مرفقه مع الكتيب ، وسنتعرض هنا أهم أجزاء في الشفرات .

لقد حاولت جعل هذه الشفرات مكتوبة بأسلوب KISS (Keep It Simple Stupid) ولكن على ما يبدوا أنها خرجت بشكل Quick And Dirty ☺. على العموم هي تعمل تماما وبدون أي أخطاء .

في جميع الشفرات التي سوف نعتمدها هنا ، سوف يكون النص الأصلي والمشفر بالحروف الكبيرة Capital Letter ، وهي الحروف التي تبدأ قيم الأسكي من 65 ، أما اذا أردت أن تتعامل مع الحروف الصغيرة فهي تبدأ من 97 . وممكن تتعامل مع النوعين ، وذلك بوضع جمله if داخل حلقة التشفير ، واطرح 65 في حال capital و97 في حال small Letter . قائمه الأسكي مرفقه في نهاية الكتيب .

يجب عليك قرائه الفصل الأول والثاني في حال أردت قراءه هذه الشفرات ، لأن أغلبها عبارة عن فكره ما ، وتطبيقها سهل جدا لمن عرف أو فهم الفكرة ، لذلك قم بقراءة الفصل الأول والثاني ، وسوف تجد الشفرات التالية في منتهى البساطة لأنها مجرد تطبيق . أو على أقل تقدير اذا كنت تبحث عن كود لأحد الشفرات ، قم بقراءتها في الفصل الثاني ، ومن ثم أقرا تطبيقها هنا .

أيضا في حالة لم تظهر الأكواد بشكل مناسب يمكنك تكبير الرؤية لتراها بوضوح أكبر .

التعامل مع الحروف *characters* :

في جميع برامج التشفير القادمة ، سوف نحتاج إلى قراءه النص الأصلي (سواء تم إدخاله مباشره من لوحه المفاتيح ، أو تم قراءته من ملف) ، ومن ثم تشفيره بالخوارزمية ، ومن ثم طباعة النص المشفر (في الشاشة أو في ملف) .عموما سنستبعد التعامل مع الملفات هنا ، لأن الغرض شرح طريقه التشفير ، وليس كيفيه إنشاء ملفات أو التعامل معها .

ولكن قبل أن نبدأ في شرح الخوارزميات ، يجب أن نختار طريقه مناسبة يمكننا عن طريقها قراءه النص المدخل من الكيبورد ، ولأن المدخل هو حروف Characters يجب أن نختار نوع بيانات مناسب . ولغة سي++ تمدنا بنوعين للتعامل مع الحروف أولها هو char والأخر string .

اذا الحل الأول ، هو أن نضع الحروف المدخلة في مصفوفة حروف array of char . والحل الثاني هو أن ندخل الحروف مباشره في النوع string (وهو الأسهل) .

لنلقى نظره على الحل الأول :

```
char plainText[10] ;

for (int i=0 ; i<=10 ; i++)
    cin>> plainText[i] ;
```

طبعا تعريف مصفوفة بخانات محدده ، أمر غير عملي ، لأنه ربما يقوم المستخدم بإدخال حروف أكثر من ذلك وبالتالي يحصل **overflow** ، وفي حاله قمت بتعريف مصفوفة كبيرة مثلا 200 خانه فسوف يكون هناك خانات فارغة سوف يكون هذا إهدار للمساحة ، لكنه أفضل من ال overflow على أية حال .

أمر أخر استخدام الكائن cin في الحلقة أمر غير جيد ، لأنه يجب أن أضغط enter بعد كل عمليه إدخال ، وهذا أمر غير جيد . لهذا سوف نستخدم داله قراءه مثل getche() ، لاحظ أن الدالة getche() تظهر الحرف عندما يتم إدخاله ، بعكس الدالة getch() التي لا تظهر الحرف .

لذلك يصبح الحل بعد استخدام الدالة getche() :

```
char plainText[10] ;

for (int i=0 ; i<10 ; i++)
    plainText[i] = getche() ;
```

بقيت مشكلتان ، أولهما حجم المصفوفة ، والأخرى أنه في المثال السابق في حال أراد المستخدم أن يدخل مثلا 3 حروف فقط ، فانه لا يستطيع لأنه محدد بحلقه تكرر 10 مرات ، يعني سيتم قراءه 10 حروف غضبن عنه ☺

.

الحل الأمثل :

```
char plainText[100] ;
char ch ;
int i= 0 ;

while ( (ch = getche()) != '\r')
     plainText[i++] = ch ;

cout << "\n\n";

for (int i=0 ; i<strlen(plainText)-1 ; i++)
     cout << plainText[i] ;
```

و هنا تكون المصفوفة ذات حجم مناسب ، ويتم القراءة حرف بحرف إلى أن يتم إدخال حرف **carriage return** والذي هو enter .

لاحظ أن الدالة strlen(array) تقوم بإرجاع طول المصفوفة . أيضا الدالة getche() معرفه داخل الهيدر #include <conio.h> لا تنسى إضافته والا سيشتكي منك المترجم .

بقيت هناك التخلص من المسافات من النص الأصلي ، وسوف نتكلم عنها هنا ولن نذكرها مره أخرى ، حيث أنك اذا أدخلت مثلا wajdy essam ، يجب أن نتخلص من المسافة والا سوف يتم تطبيق داله التشفير عليها ، طبعا يمكنك تركها ، ويمكنك إلغائها و هذا ما أفضله .

```
while ( (ch = getche()) != '\r')
 {
       if ( ch == ' ' )
            continue ;

       plainText[i++] = ch ;
 }
```

لاحظ تم اختبار الحرف هل هو المسافة ، فإذا كان كذلك فيتم الرجوع إلى الحلقة مره أخرى .

حل أخر بدون استخدام الدالة getch() ، واستخدام الدالة cin.get(array,size,endchar):

```
char plainText[100] ;

cin.get(plainText,100,'$');

for (int i=0 ; i<strlen(plainText) ; i++)
     cout << plainText[i] ;
```

قد يبدوا الحل مناسبا ، ولكن في حال أردت قرائه متغير ما بعد هذه الجملة cin.get(plaintext,100,'$') فلن تستطيع ، والسبب يعود إلى أن الحرف $ سوف يحل مكان القيمة التي تريد قراءتها ، لكي تفهم أكثر ، **قم بتشغيل المثال القادم** :

```
char plainText[100] ;

cin.get(plainText,100,'$');

// you cann't read this variable
 int x  ;
cin >> x ;

for (int i=0 ; i<strlen(plainText) ; i++)
    cout << plainText[i] ;
```

حل هذه المشكلة هو استخدام الدالة cin.getline(plaintext,100,'$') ، وسوف تستطيع قرائه متغيرك بسهوله :

```
char plainText[100] ;

cin.getline(plainText,100,'$');

//Now you can :)
 int x  ;
cin >> x ;

for (int i=0 ; i<strlen(plainText) ; i++)
    cout << plainText[i] ;
```

نأتي الآن إلى التعامل مع string وهو ما سأعتمده لكتابه الشفرات ، نظرا لسهولته ، وكل شيء يكون تلقائي .

اذا استخدمت الدالة (في الحقيقة هي **object**) cin لقراءة متغير من نوع string فلن تستطيع استخدام المسافة ، جرب البرنامج التالي ، وأدخل wajdy essam ، وسوف تلاحظ المخرج wajdy لان cin تعتبر المسافة هي نهاية النص .

```
string name ;
cin >> name ;

cout << name ;
```

ولكن بقليل من التحايل ، نستطيع فعل ما نريد وبعده طرق ، وأفضل دائما استخدام الدالة getche() لأنها لا تحتاج لضغط enter بعد عمليه الإدخال .

<u>بهذه الطريقه القادمة ، سوف نقوم بقراءة جميع النصوص المراد تشفيرها فى خوارزميات التشفير (هى ليست الطريقة الأفضل ، لكنها تؤدى الغرض) .</u>

```
string str ;
char ch ;

while ( (ch=getche()) != '\r'   )
{
        if ( ch == ' ' )
            continue ;

        str += ch ;
}
```

وفي حال أردنا معالجه الstring في حلقه ، فيكون التعامل معها كما هو الحال مع المصفوفة str[i] أي باستخدام index .

نأتي الآن إلى خوارزميات الفصل الأول وهي الخوارزميات الرياضية :

إيجاد القاسم المشترك الأعظم *Greatest Common Divisor* تم كتابه أكثر من طريقه لتطبيق هذه الخوارزمية ، ويمكنك استخدام ما تشاء ،

الحل الأول ، وهو بطريقه عاديه Classical :

```
int Classical_GCD (int x ,int y)
{
     int small = 0;

     if ( x < y )
          small = x ;
     else
          small = y ;

     while ( x%small != 0 || y%small != 0 )
          small-- ;

     return small ;
}
```

وهنا طريقته بعد معرفه الأصغر من بين عددين يتم اختبار باقي قسمه كل عدد من هذين العددين مع العدد الصغير ، فإذا كان باقي القسمة للعدين في نفس اللحظة يساوي 0 ، معناها وصلنا للقاسم المشترك الأعظم ، والا نطرح من العدد واحد ونعيد الكره مره أخرى .

حل أخر ، بواسطة خوارزمية أقليدس ، وتم كتابه ثلاثة implementation لها ، الأول والثاني متشابهان ، والثالث تم استخدام مفهوم النداء الذاتي **Recursion Function**

```
int Euclid_GCD (int x ,int y)
{
     int tmp = 0 ;

     while ( x > 0 )
     {
          if ( x < y )
          {
               tmp = x ;
               x = y ;
               y = tmp ;
          }

          x = x-y ;
     }

     return y ;
}
```

الحل بواسطة النداء الذاتي :

```
int Euclid_GCD_Recursion (int x, int y)
{
     if ( y == 0 )
          return x ;

     else
          return Euclid_GCD_Recursion(y,x%y);
}
```

خوارزمية اقليدس الممتدة

أولا القوانين هي :

```
* Extended Euclid's Algorithms
* it's use the Flowing Rules :
     s0 = 1 , s1 = 0 , si = si-2 - (si-1 * qi-1)
     t0 = 0 , t1 = 1 ,  ti = ti-2 - (ti-1 * qi-1)

     r0 = first number
     r1 = second number
     r1 = r-2 % r-1  (the first last number MOD the second last Number)


* The GCD (a,b) = s.a + t.b
```

والحل هو :

```
// save this number in varibale
int r0 = num1 ;
int r1 = num2 ;
// q1 is the qoution
int q = r0 / r1 ;
int r2 = r0 % r1 ;
int s2 , t2 ;
while ( r2 > 0 )
{
        s2 = s0 - (s1 * q) ;
        s0 = s1 ;
        s1 = s2 ;

        t2 = t0 - (t1 * q) ;
        t0 = t1 ;
        t1 = t2 ;

        r0 = r1;
        r1 = r2 ;

        q = r0 / r1 ;
        r2 = r0 % r1 ;
}
```

اختبار أوليه العدد باستخدام *Trial Division*

```
bool isPrime (int n )
{
    for (int i=2 ; i<= sqrt(n) ; i++)
        if ( n%i == 0 )
            return false;

    // it's prime number
    return true;
}
```

التعامل مع اللوغريتم

```
double log10 (double x)
{
    double r = log(x) / log(10.0);
    return r ;
}

double log2 (double x)
{
    double r = log(x) / log(2.0) ;
    return r ;
}

double log10Tolog2 (double log10 )
{
    double r = log10 * 3.322 ;
    return r ;
}
```

Fast Exponentiation Algorithms

إيجاد x^y عندما يكون y كبيرا جدا :

و هذه هي الطريقه التقليدية ، حيث نعمل حلقه بعدد y ومن ثم نضرب x في نفسه ونخزن الناتج في متغير أخر .
النوع __int64 ، هو نفسه long long int ومدى الأعداد به كبير ، 2^64 -1 .

ولكن هذه الطريقه غير جيده ، وخاصة في حال كان y كبير ، لأننا سوف نمر بعدد y مره ، وهو شيء غير عملي اذا كان مثلا 2^1024 .

```
__int64 Simple_EXP (int x , int y)
{
    __int64 p = 1 ;

    for (int i=0 ; i<y ; i++)
        p *= x ;

    return p ;
}
```

وهذه الطريقه الأفضل :

```
__int64 Fast_EXP (int x ,int y)
{
     __int64 p = 1 ;

     while ( y > 0 )
     {
          while ( y % 2 == 0 )
          {
               x = x * x ;
               y = y / 2 ;
          }

          p *= x ;
          y--;
     }

     return p ;
}
```

نختتم برامجنا الرياضية ، ببرنامج **التحويل بين أنظمه الأعداد** ، hex , binary , decimal , octal ، وهذه تصريح الدوال ، علما ، بأن البرنامج موجود مع البرامج المرفقة ، ويمكنك الرجوع إليه لقرائه تعريف الدوال .

```
void decimalToBinary (int dec       ) ;
void decimalToHex    (int dec       ) ;
void decimalToOctal  (int dec       ) ;

int  binaryToDecimal (char binary[] ) ;
int  hexToDecimal    (char hex[]    ) ;
int  octalToDecimal  (char octal[]  ) ;


void binaryToHex     (char binary[] ) ;
void binaryToOctal   (char binary[] ) ;

void hexToBinary     (char hex[]    ) ;
void hexToOctal      (char hex[]    ) ;

void octalToHex      (char octal[]  ) ;
void octalToBinary   (char octal[]  ) ;
```

البرنامج "غير ضروري" في كتيبنا هذا ، لكن وضعنها من باب الفائدة .

شفرة قيصر *Caesar Cipher*

كما رأينا في شفرة قيصر يكون المفتاح (مقدار الإزاحة) هو 3 . وفي حال تعدى الحرف الناتج عن أخر حرف في الأبجدية فيجب أن نرجع إلى الحرف الأول .
كيف يمكن برمجه ذلك ؟
هناك عده حلول ، ولكني أفضل حلين منها ، وهما :

الحل الأول :
عمل مصفوفتين ، الأولى للحروف العادية (النص الأصلي) والأخرى للنص المشفر الناتج من خوارزمية قيصر ، الأولى نقوم بتهيئتها بالحروف من A إلى Z ، والثانية نهيئها بحروف من D إلى Z ثم A,B,C .

```
char normalChar[26] ; // contain the character alphabetic form A-Z
char cipherChar[26] ; // container character from D-Z+A,B,C
```

الآن عمليه التهيئة :

```
// initializing normalChar and cipherChar
int i ;
for (char ch='A' , i=0  ; i<26 ; i++,ch++)
```

```
        normalChar[i] = ch ;

    for (i=0 ; i<26 ; i++)
        cipherChar[i] = normalChar[(i+3)%26];
```

الآن ، لكي نقوم بالتشفير ، نفرض أننا أدخلنا النص المراد تشفيره في المصفوفة inputChar ، ونريد النص المشفر يكون في المصفوفة outputChar :

```
 for (i=0 ; i<length ; i++)
 {
     if ( inputChar[i] == ' ' )           // if it's space , ignore it
     {
             outputChar[i] = ' ' ;
             continue;
     }

     int x = (int)inputChar[i] ;          // take the character as number

     x = x -  65 ;                        // make the number from 0-25

 outputChar[i] = cipherChar[x] ;   // assign the cipher char to output
     cout << outputChar[i]   ;
 }
```

داخل الحلقة ، قمنا بتحويل الحرف الأول في المصفوفة إلى نوع int ، وهكذا تحصلنا على قيمه الأسكي الخاص بالحرف الأول ، ولنفرض أنه الحرف B فقيمه الأسكي هنا في هذه الحالة 66 . بعدها نطرح من هذه القيمة 65 ، لكي نرجع قيمه المتغير إلى رقم مداه من 0 إلى 25 ، وهنا يكون هذا المتغير هو الindex الخاص بمصفوفة الحروف المشفرة cipherChar التي قمنا بتهيئتها في البداية .

لفك التشفير ، لنفرض أن النص المشفر موجود داخل outputChar ونريد النص الأصلي في newNormal

```
for (int i=0 ; i<length ; i++)
{
    if ( outputChar[i] == ' ' )
    {
         newNormal[i] = ' ' ;
         continue;
    }

    int x = (int)outputChar[i] ;  // get cipher char , convert to int

    x = x - 65 ;                    // return it to number
    x = x - key ;                  // sub from it the key

    // if it's negative make it from the begin of array
    if ( x<0)   x = 26-abs(x) ;

    newNormal[i] = normalChar[x] ;
    cout << newNormal[i] ;
}
```

الحل الثاني والذي أفضله ، للتشفير نقوم بالتالي :

```
for (int i=0 ; i<str.length() ; i++) // str contain the plainText
{
    int x = ( ( ((int)str[i] - 65) + KEY ) % 26 ) + 65 ;
    str2 += (char) x ;   // str2 contain the cipher text
}
```

يهمنا هنا السطر الأول ، لأننا سنستخدمه كثيرا في الشفرات القادمة ، ونبدأ بشرحه من داخل الأقواس ، أولا نأخذ الحرف الأول من النص الأصلي ونقوم بتحويله إلى int لأخذ قيمه الأسكي الخاص بالحرف ، بعدها نطرح منه 65 لكي نحصر القيمة من 0 إلى 25 ، الآن نجمع إليها المفتاح ، وهو هنا في خوارزمية قيصر 3 ، ونأخذ باقي القسمة الناتج على 26 ، بعدها نرجع الحرف إلى قيمته وذلك بجمع 65 إليه . أخيرا نسند هذه القيمة بعد تحويلها إلى حرف إلى النص المشفر .

ولفك التشفير ، نقوم بالتالي :

```
for (int i=0 ; i<str.length() ; i++)
{
    int x =  ( ( ((int)str[i] - 65) - KEY ) % 26 )  ;
    x = (x<0) ? (26- abs(x)) : x ;
    str2 += (char)x+65 ;
}
```

و هي مشابه لطريقة التشفير ما عدا في السطر الأول نقوم بطرح المفتاح من الحرف الأول ، أما السطر الثاني ، فهو يتأكد من القيمة السابقة (x) أكبر من 0 ،و إلا في هذه الحالة أطرح 26 – القيمة المطلقة لهذا الحرف ، ولنعرف الغرض منه نأخذ المثال التالي :
الحرف E نحوله إلى قيمه الأسكي لتخرج إلينا الرقم 69 ، الآن نطرح منه 65 ليصبح لدينا 4 ، ونطرح منها المفتاح الذي هو 3 ، ليخرج لدينا 1 ، نأخذ 1 باقي قسمه 26 ليخرج واحد نفسه . الآن السطر الثاني لا يهم ، لأن القيمة أكبر من 0 .

نأخذ الحرف A ، نحوله إلى قيمه الأسكي لتخرج لدينا القيمة 65 نطرح منها 65 يطلع لدينا 0 ، نطرح منها المفتاح 3 ليخرج لدينا -3 ، الآن ننفذ السطر الثاني ونطرح 26-3 = 23 وقيمته هي الحرف X و هو الحرف الصحيح .

شفرة ROT13

بنفس طريقه شفره قيصر ، لكن هنا المفتاح يكون 13 ، وداله التشفير ، هي نفسها التي تقوم بفك التشفير .

```
for (int i=0 ; i< str.length() ; i++) // key = 13
{
    int x = ( ( ((int)str[i] - 65) + KEY ) % 26 ) + 65 ;
    str2 += (char) x ;
}
```

التشفير بطريقه Affine Cipher

القوانين :

```
/* to Encrypt With Affine Cipher :
       c = m * p + key (mod n)          where m is Multiplirer , p is character in plaintext , key is shift number , n is alpha size
  Note : to choose the right m , it must the GCD(m,n) = 1 , i.e m and n is relative prime number , i.e must use GCD method to check

  To Decryption with Affine Cipher :
       p = m` * ( c - key) (mod n)       where m` is the inverse of m
  Note : to obtain the m`  it must use Extended Euclid's Method
```

الآن لمعرفه هل m و n أوليان فيما بينهما ، يجب أن يكون القاسم المشترك الأعظم لهما يساوي واحد ، فنقوم باستخدام داله GCD لإيجاد القاسم (والتي قد ذكرناها سابقا) ، ونقارن الناتج هل يساوي واحد ، اذا كان كذلك هم أوليان فيما بينهما ، والا هم غير ذلك .

```
bool AffineCipher :: checkKey (int m ,int n)
{
     if ( GCD(m,n) == 1 )
          return true ;

     else
          return false;
}
```

ونطبق قانون التشفير :

```
string AffineCipher :: encryption (string plainText  , int m , int key , int n )
{
    string str2 = "" ;

    for (int i=0 ; i<plainText.length() ; i++)
    {
        int x = ( ( ( ((int)plainText[i] - 65) * m)  + key ) % n ) + 65 ;
        str2 += (char) x ;
    }
                            c = p * m + key % n  نطبق القانون
    return str2 ;
}
```

ولفك التشفير ، نستدعي داله extended Euclid خوارزمية اقليدس الممتدة ، لإيجاد المعكوس ، **وبعدها** :

```
string AffineCipher :: decryption (string cipherText , int m , int key , int n )
{
    string str2 = "" ;
    int inverse_M = getInverse(m,n) ;

    for (int i=0 ; i<cipherText.length() ; i++)
    {
        int x =  ( ( inverse_M * ( ((int)cipherText[i] - 65) - key )  ) % n )  ;
        str2 += (char)x+65 ;
    }

    return str2 ;
}
```

هنا سيرجع معكوس m

نطبق قانون فك التشفير $c = m` * (c - key) \% n$

شفرات عائله Vigenere :

هذه الشفرات متشابه فيما بينهما بشكل كبير (أقصد من ناحية برمجيه) ، لذلك سوف يكون من السهل تتبع هذه الشفرات ، وخاصة اذا كنت تعرف طريقه عمل كل شفره (أي قمت بقراءة الفصل الثاني) .

شفرة فيجنير البسيطة *Simple Shift Vigenere Cipher*

للتشفير نقوم بجمع المفتاح مع النص الأصلي % 26 (بالضبط كما هو الحال مع شفره قيصر) ، لكن في حال انتهي المفتاح سوف يتكرر (أي يرجع من البداية) ، وهنا تكون الفكرة ، بأن أعرف طول المفتاح ، بعدها أقوم بحصر مدى المفتاح (أي يكون من البداية وحتى النهاية ، وفي حال تكرر يرجع من البداية) ويمكن عمل ذلك عن طريق معامل باقي القسمة ، كما هو المثال التالي :

والباقي كما هو في خورزميه قيصر ، الآن لفك التشفير :

```
string SimpleVigenere :: Encryption (string plainText  , string keyPhrase )
{
    string cipherText = "";
    int kplength = keyPhrase.length();

    for (int i=0 ; i<plainText.length() ; i++)
    {
        int x = ( ((int) plainText[i]) - 65 ) + ( ((int) keyPhrase[i%kplength]) - 65 );
        x = x % 26 ;
        cipherText += (char) x+65 ;
    }

    return cipherText ;
}
```

هنا نقوم بحساب طول المفتاح

شفره فيجينر الكاملة *Full Vigenere Cipher*

و هنا يجب في البداية أن ننشئ جدول التشفير a tabular recta ، ويمكن إنشائه يدويا عن طريق مصفوفة من بعدين ونقوم بتعبئتها بجميع الحروف كم في الجدول ، ولكنه حل متعب ويفضل استخدام الحلقة ، **الجدول كما في الشكل التالي (وقد سبق عرضه في الفصل الثاني) :**

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

عمل الجدول :

```
void  FullVigenere  :: makeVigenereTable ()
{
    char ch = 'A' ;

    for (int i=0 ; i<26 ; i++)
    {
        for (int j=0 ; j<26 ; j++)
        {
            int x = (((int)ch-65)+j)%26;
            vigenereTable[i][j] = (char)x+65 ;
        }

        ch++;
    }

}
```

الآن للتشفير نأخذ نقطه تقاطع الحرف من النص الأصلي ومن المفتاح ، (أكرر :راجع الفصل الثاني ، حتى تستطيع فهم الطريقه) . **والدالة التى ترجع نقطه التقاطع هي** :

```
char  FullVigenere :: getCharFromVigenereTable (int col , int row )
{
    return vigenereTable[row][col] ;
}
```

الآن للتشفير :

```
string FullVigenere :: Encryption (string plainText  , string keyPhrase )
{
    string cipherText = "";
    int kplength = keyPhrase.length();

    for (int i=0 ; i<plainText.length() ; i++)
    {
        int x = ( ((int) plainText[i]) - 65 ) ;
        int y = ( ((int) keyPhrase[i%kplength]) - 65 );

        cipherText += getCharFromVigenereTable(x,y) ;
    }

    return cipherText ;
}
```

ترجع نقطه التقاطع ليكون هو النص المشفر

ولفك التشفير :

```
string FullVigenere ::Decryption (string cipherText , string keyPhrase )
{
    string plainText = "";
    int kplength = keyPhrase.length();

    for (int i=0 ; i<cipherText.length() ; i++)
    {
        int x = ( ((int) cipherText[i]) - 65 ) ;
        int y = ( ((int) keyPhrase[i%kplength]) - 65 );

        char ch = cipherText[i] ;
        int j ;
        for ( j=0 ; j<26 ; j++)
        {
            if ( getCharFromVigenereTable(y,j) == ch )
             break;
        }

        plainText += getCharFromVigenereTable(0,j) ;
    }

    return plainText ;
}
```

حصلنا على الحرف المشفر

نقوم بالبحث في المصفوفه عن موقع المفتاح وموقع ما يعطونا الحرف المشفر ، ونرجع هذا الموقع المجهول j

اذا الحرف الأصلي هو العمود j في الصف الأول 0

شفره فجينير تلقائية المفتاح *Auto Key Vigenere Cipher*

وفي هذه الشفره وبعد انتهاء المفتاح ، يدخل المفتاح الأصلي في عمليه التشفير ، أي يصبح هو المفتاح . وفي عمليه فك التشفير وبعد انتهاء المفتاح ، يكون المفتاح الحالي هو أول حرف تم فك تشفيره ، والمفتاح التالي ، هو ثاني حرف فك تشفيره وهكذا (راجع الفصل الثاني ، للمزيد من الأمثلة والتوضيح) .

الآن وقبل البدء بالتشفير ، الفكرة هنا، أن نقارن المفتاح بالنص الأصلي ، فإذا كانا نفس الطول كان بها ، والا قم بجمع الحرف الأول من النص الأصلي بعد أخر حرف في المفتاح ، واجمع الحرف الثاني من النص الأصلي بعد ثاني أخر حرف في المفتاح ، واستمر هكذا إلى أن يصبح النص الأصلي يساوي المفتاح .

```
int kplength = keyPhrase.length();
int ptlength = plainText.length();

if ( kplength < ptlength)
{
     int x = ptlength - kplength ;

     for( int i=0 ; i<x ; i++)
          keyPhrase += plainText[i]  ;
}
```

الآن التشفير سوف يكون بالطريقة التقليدية كما في شفره قيصر ، لان المفتاح حاليا يساوي النص الأصلي :

```
string AutoKeyVigenere :: Encryption (string plainText  , string keyPhrase )
{
    string cipherText = "";

    for (int i=0 ; i<plainText.length() ; i++)
    {
        int x = ( ((int) plainText[i]) - 65 ) + ( ((int) keyPhrase[i]) - 65 );
        x = x % 26 ;
        cipherText += (char) x+65 ;
    }

    return cipherText ;
}
```

ولفك التشفير ، في كل خطوه نقوم بجمع النص الأصلي (الذي تم فك تشفيره في هذه اللحظة) إلى أخر المفتاح :

```
string AutoKeyVigenere ::Decryption (string cipherText , string keyPhrase )
{
    string plainText = "";

    for (int i=0 ; i<cipherText.length() ; i++)
    {
        int x = ( ((int) cipherText[i]) - 65 ) - ( ((int) keyPhrase[i]) - 65 );
        x = x % 26 ;
        x = (x<0) ? (26- abs(x)) : x ;
        plainText += (char) x+65 ;
        keyPhrase += (char) x+65 ;    // add the plainText at end of key
    }

    return plainText ;
}
```

شفره فيجينر طويلة المفتاح *the Running key Vigenere Cipher*

هنا يجب أن نتحقق من الشرط قبل البدء في التشفير ، وهو أن المفتاح يكون أطول من النص العادي (أو يساويه) ،. وتكون عمليه التشفير وفك التشفير عاديه كما في الأمثلة السابقة .

هنا مثال يبين كيفيه التحقق ، وفي حال كان طول المفتاح أقل ، يتم الخروج ولا تتم عملية التشفير :

```
int kplength = keyPhrase.length();
int ptlength = plainText.length();

if ( kplength < ptlength )
{
     cout << "\n\nKeyPhrase Must me Greater or same length as plain Text
     break ;
}
```

شفرة بلافير Playfair Cipher

شفره بلافير أخذت مني الكثير من الأسطر، لذلك لن أضعها هنا ، ويمكنك مشاهده الكود مرفق مع الكتاب ، علما بأن تطبيقها سهل ،اضافه إلى أني قمت بكتابة الكثير من التعليقات التي سوف توضح عمل كل داله . وبإذن الله تكون واضحة للجميع .

شفره Reverse Cipher

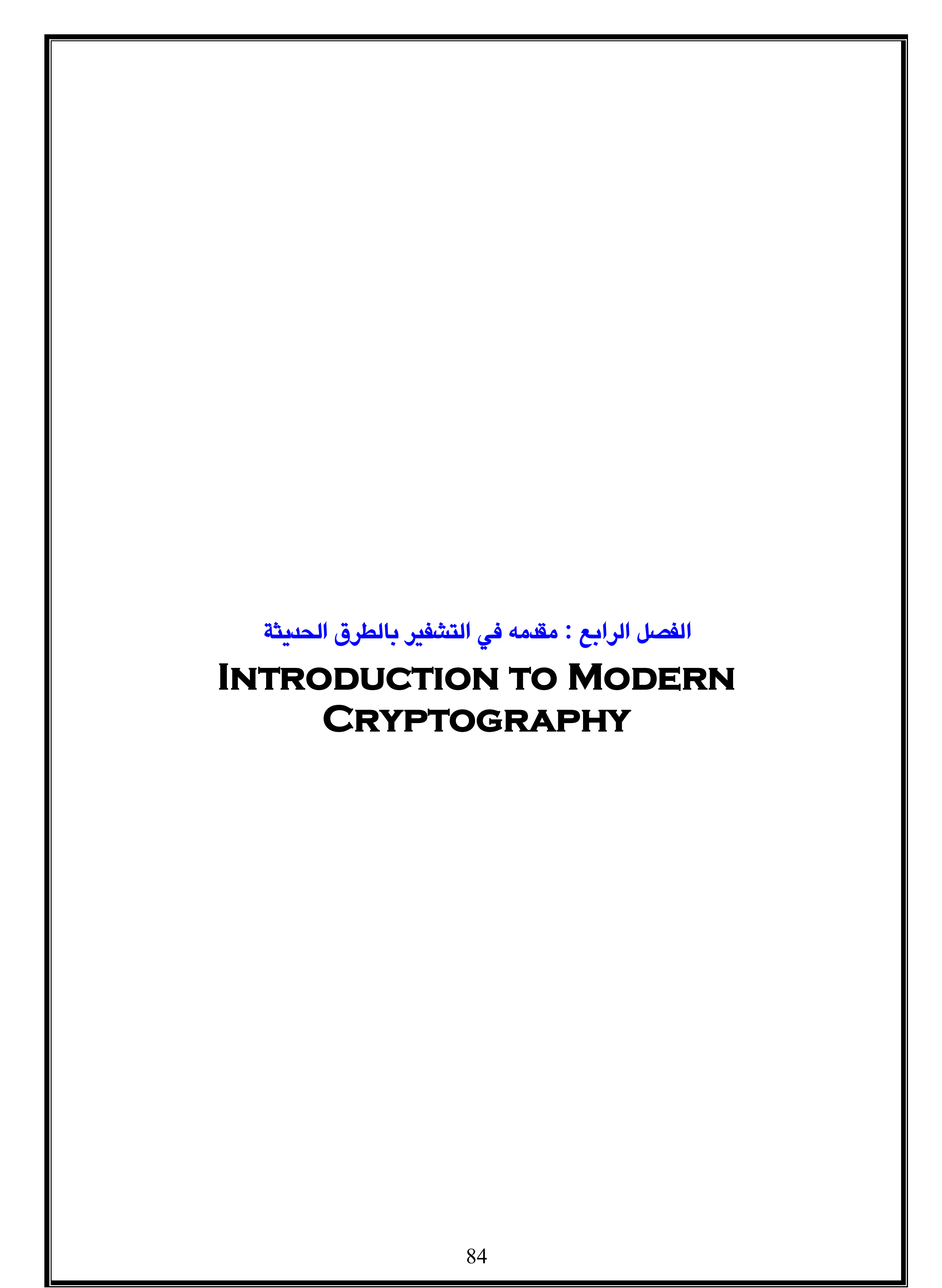
هنا بعد إدخال النص نقوم بطباعته معكوس (من النهاية إلى البداية) ، لكن لاحظ هنا استخدمنا طباعه كل 5 حروف في بلوك (أي تفصل بينهم مسافة) وهو أمر مفروض أن يكون في جميع الشفرات السابقة ، ويكون عليك تطبيقها كواجب منزلي ☺

```
int cx = 1 ;

for (int i=str.length()-1 ; i>=0 ; i--, cx++)
{
    if ( cx % 5 == 1)
       cout << "  " ;

    cout << str[i] ;

}
```



الفصل الرابع : مقدمه في التشفير بالطرق الحديثة

Introduction to Modern Cryptography

لماذا بالتشفير *Why Cryptography* ؟

قد تتساءل أخي الكريم وتقول " أنا لا احتاج لأي نوع من التشفير ، ببساطه ، لأنه ليس لدى أي شيء لأخفيه ، ولا داعي لهذه الحماية" ؟

حسنا أخي ، لكن دعني أشاهد ملفاتك الطبية ، أو كشف حساباتك في احد البنوك ، أو حافظه نقودك ، أو دعني استخدم رقم(الهوية) الجواز الخاص بك ، أو على اقل تقدير دعني استخدم الباسورد الخاص بدخول المنتدى ، أو البريد الخاص بك .

المقصد هنا ، أننا جميعا لدينا معلومات بحاجه إلى أن تبقى سريه عن الجميع ، بالتأكيد ستشعر بعدم الارتياح مثلا في حاله عرفنا كلنا بمواعيد زياراتك للطبيب ، ما هي الادويه والعلاجات التي تستخدمها ، ما هي الأمراض المصاب بها (عافانا الله وإياكم) .. سبب أخر نحن نريد الحفاظ على تلك المعلومات من المخترقين ، تخيل احدهم سرق الباسورد الخاص بك في المنتدى ودخل بحسابك وبدا يسب ويعلن في الأعضاء ماذا سيكون موقفك ، بالتأكيد أهون من الذي تم استخدام بطاقته الائتمانية وسرقه مبلغ 1000$ ☺ .

الشركات أيضا لديها العديد من الأسرار (الاستراتيجيات ، تفاصيل المنتجات ، معلومات الموظفين ، نتائج أبحاث سريه) ، وحتى الشركات النصابة "وهم كثر" تريد الحفاظ على هذه النشاطات الدنيئة بعيدا عن الأعين ، كل هذه الشركات (بغض النظر عن شرعيتها) تريد أن تحفظ معلوماتها بعيدا الناس (قد يكونوا منافسين ، مخترقين "هكر أم كراكر") .

الحماية المقدمة من قبل نظام التشغيل :

في الماضي كانت عمليه الحماية بسيطة جدا ، فقط كل ما عليك هو أن تضع ملفاتك في الدرج ثم إغلاق باب المكتب بالمفتاح !! وإذا كنت تريد حماية أكثر يمكنك الإستعانه "بطبله كبيرة"وانتهى الأمر ، لكن حاليا الأمر اختلف بشكل كبير ، فالملفات أصبحت تحفظ في الجهاز والهارديسك والسي دي CD و وسائل أخرى غيرها ، السؤال هو كيف يمكن حماية الهارديسك ؟

وهنا يأتي دور نظام التشغيل، اغلب نظم التشغيل تقدم نوع من الحماية وتسمى الصلاحيات **permissions** والتي تسمح لمستخدمين معينين الدخول إلى النظام، وذلك عن طريق إجراء الدخول (نافذة login) ، وفي حاله إدخال اسم المستخدم والباسورد الصحيحين يتم الدخول إلى النظام ، وحتى بعد الدخول في حاله قم بحذف ملف معين فإن نظام التشغيل ينظر أولا هل آنت من لك الصلاحية لحذف هذا الملف وفي حاله النفي فانك لن تستطيع حذف الملف أبدا(طبعا مستخدمين انظمه Unix-Like يعرفوا هذه التفاصيل بشكل كبير)

والشخص الذي يكون لديه الصلاحية لفعل شيء هو المدير administrator أو root ، وهو الذي يعطى الصلاحيات لباقي المستخدمين ، هو الذي يعطل الحسابات ، هو الذي يعطي الباسورد في حال فقد احدهم الباسورد الخاص به ، هو المدير بمعنى الكلمة .

كيف يعرف نظام التشغيل أن الشخص الموجود هو المدير الفعلي ؟ عن طريق اسم المستخدم و الباسورد بالطبع، لكن وبكل أسف الطرق لمراوغه النظام وكسر حماية باسورد المدير أصبحت منتشرة وبكثرة، في ويندوز xp home يكفي أن تدخل عن طريق **الوضع الأمن safe Mode** للدخول إلى النظام !! وهناك الكثير من البرامج لكسر وتغيير الباسورد للأغلب الانظمه..

كمخترق أريد أن ادخل إلى النظام أول ما أفكر فيه هو اسم المستخدم الذي يأتي مع النظام **preset account**، بالاضافه إلى أن اغلب أنظمه ويندوز يكون فيها اسم المستخدم الخاص بالمدير هو **administrator** وبدون باسورد، وللأسف الكثير من المستخدمين يتركون هذا الحساب بدون تغيير، وفي هذه الحالة الدخول إلى النظام أمر في منتهى البساطة وسوف اشكر الشركة المنتجة على هذه لصنيعه التي لا تنسى ☺ .

للأسف المستخدم طلع عامل حسابه ومغير الباسورد، اذا سأبحث عن طريق أخر وهو البحث عن اسم المستخدم والباسورد يعني تخمينه، في الأفلام والمسلسلات الأمر سهل للغاية، دقيقه أو اثنين ويتم الكشف عن هذا الباسورد ويتم الدخول إلى النظام، لكن الحقيقة تختلف كثيرا، ربما اذا بحثت في المكتب تجد بعض الأوراق مكتوب عليها الباسورد (أيضا هذه في الأفلام فقط) .

حسنا، على العموم هناك طريقه تستخدمها الكثير من الجامعات والمؤسسات وهي اسم المستخدم هو نفسه اسم الباسورد، وقد حصل هذا الأمر معي في الجامعة ودخلت إلى النظام، كان اسم المستخدم والباسورد عبارة عن اسم الشركة المصنعة للشاشة، مكونه من ثلاث حروف، طبعا بعد العديد من المحاولات البائت بالفشل، ولم انتبه إلى انه الباسورد مكتوب في أسفل الشاشة.

حسنا ، في حال اسم المستخدم لم يكن هو الباسورد ، ماذا افعل ؟ مع الأخذ بالاعتبار إني اعرف اسم المستخدم ، لا توجد طريقه إلا بالتخمين حول الباسورد ، اسم الزوجة ، رقم الهاتف ، تاريخ الميلاد ، اسم الحبيبة ، مممم مشكله أليس كذلك ! بالطبع لا ، فهناك الكثير من البرامج تقوم بعمليات التخمين نيابة عنك **password cracker** ، وفي حاله الباسورد ضعيف ، سوف تدخل إلى النظام في خلال دقائق .

هناك برنامج اسمه **l0phtCrack** يستخدم من قبل مدراء الانظمه ، وظيفته تغيير الباسودرات المستخدمة في الشبكة ، (بالطبع في حال المدير يستطيع ذلك ، الكراكر يفعلها أيضا)

نوع أخر من أنواع الهجوم هو تجاوز نظام التشغيل ، وهو يتطلب بعض الخبرة في هذا المجال ، مثلا Data Recovery Attack ، وهنا سوف يقوم بقراءة الهارديسك بت بت وتجميعها لبناء الملف الأصلي ، وهذه البرامج الغرض منها ليس للهجوم ولكن الكراكرز هم الذين استفادوا منها فاغلب برامج استعاده البيانات يتم استخدامها من قبل المختصين في استرجاع البيانات ، مثلا خرب عليك النظام System Crash أو حدث Bad Sector في الهادريسك الخاص بك ، كل ما عليك (في حال انك مستخدم) الذهاب إلى خبراء استرجاع البيانات ، وهو سوف يستخدموا هذه البرامج لاستعاده بياناتك . نفس الأمر سوف يقوم المخترق باستخدام هذه البرامج لتجاوز نظام التشغيل..

نوع أخر من الهجوم وهو الهجوم على الذاكرة Memory Reconstruction Attack ، في البداية عندما نتعامل مع برنامج ما بالطبع سوف تكون جميع التعليمات موجودة في الذاكرة ، وسوف تكون هناك اشاره في الموقع المحفوظ فيها تعليمات البرنامج ، وعندما ننتهي من البرنامج ونقوم بإغلاقه ، فسوف يقوم مدير الذاكرة في النظام بحذف هذه الاشاره دون حذف المحتوى الحقيقي لها ، بالطبع من الممكن أن يأتي أي برنامج أخر ويحل في نفس الموقع ويحذف تلك البيانات ، ومن الممكن أن تكون موجودة .. هجوم الذاكرة يقوم بعمل مسح للذاكرة وكتابه تلك البيانات التي لا توجد عليها اشاره ، (لا اعلم كم يتم تطبيق هذا الهجوم على ارض الواقع) .

مشكله أخرى ، وهي الذاكرة الظاهرية ، اغلب انظمه التشغيل تحتوي على Virtual Memory

وهنا يتم استخدام الهادريسك كذاكره ، ومبدأ عملها في حاله امتلأت الذاكرة يتم إفراغ بعض من محتوياتها التي لم تستخدم من وقت طويل إلى الهادريسك في مكان معين ، بعدها في حال طلبت تلك المواقع (التي أصبحت موجودة في الهارديسك) سوف يقوم نظام الذاكرة بعمل تبديل Swap بين المحتويات ، على العموم في عام 1999 قام احدهم بكتابه برنامج لمسح هذا المكان التي تحفظ فيه الذاكرة وتمكن من إيجاد الباسورد الخاص به في احد البرامج.

كل طرق الهجوم السابقة ، دليل على أنك وحتى "بنظام تشغيلك الخارق" قد تكون بيانات وملفاتك في خطر ، لذلك لا بد من اضافه الحماية بواسطة التشفير و عدم الاعتماد على تلك الحماية المقدمة من نظام التشغيل والافتراض بأن المخترق يعرف أساليب التراوغ والاختراق ، والتشفير هو ببساطه تحويل النصوص المفهومة غلى كلام غير مفهوم gibberish ،

مثل: my name is wajdy , im a Beginer in java programming
تصبح: kjdkp isjeu epdmp owdkl kld dkl kqklq ds

وحتى لو استطاع المخترق بالوصول إلى نظامك وكسره ، سوف يشاهد ملفك بالصورة السابقة ، ولن يحصل على شيء مفيد أبدا .

يعني بالتشفير سوف تحصل على (أهداف التشفير):

*الخصوصية أو السرية Privacy
لن يستطيع احد قرائه ملفاتك السرية (وملفاتك الطبية) ، إلا من تريده أنت فقط !

*تكامل البيانات Data Integrity
ويعني التأكد من أن رسالتك لم تتغير (قام احدهم بتغيير شيء ما) أثناء إرسالك للرسالة ، أو قام بتغيير ملف محفوظ مسبقا .

*التحقق Authentication
التحقق من الشخص الفلاني هو الشخص الذي تريده لقرائه الرسالة ،

*عدم الإنكار nonrepudiation :
مصطلح غريب قليلا ، ولكن الفائدة هنا جعل الشخص المرسل للرسالة الالتزام وعدم إنكار انه هو الشخص المرسل للرسالة .

التشفير بالمفتاح المتناظر *Symmetric key Cryptography* :

نذكر مره أخرى أن التشفير هو عبارة عن تحويل المعلومات المفهومة إلى معلومات غير مفهومه Gibberish ، والعملية العكسية فك التشفير ، هي عمليه تحويل المعلومات الغير مفهومه إلى معلومات مفهومه .

النوع الأول من أنواع التشفير هو: التشفير بالمفتاح المتناظر ، وهنا سوف نستخدم مفتاح مع خوارزمية (هناك الكثير) لتشفير المعلومات ، وسوف نستخدم نفس المفتاح ونفس الخوارزمية لفك التشفير . (لاحظ يجب أن تكون نفس المفتاح ونفس الخوارزمية ، ومن هنا جاء الاسم "متناظر") .

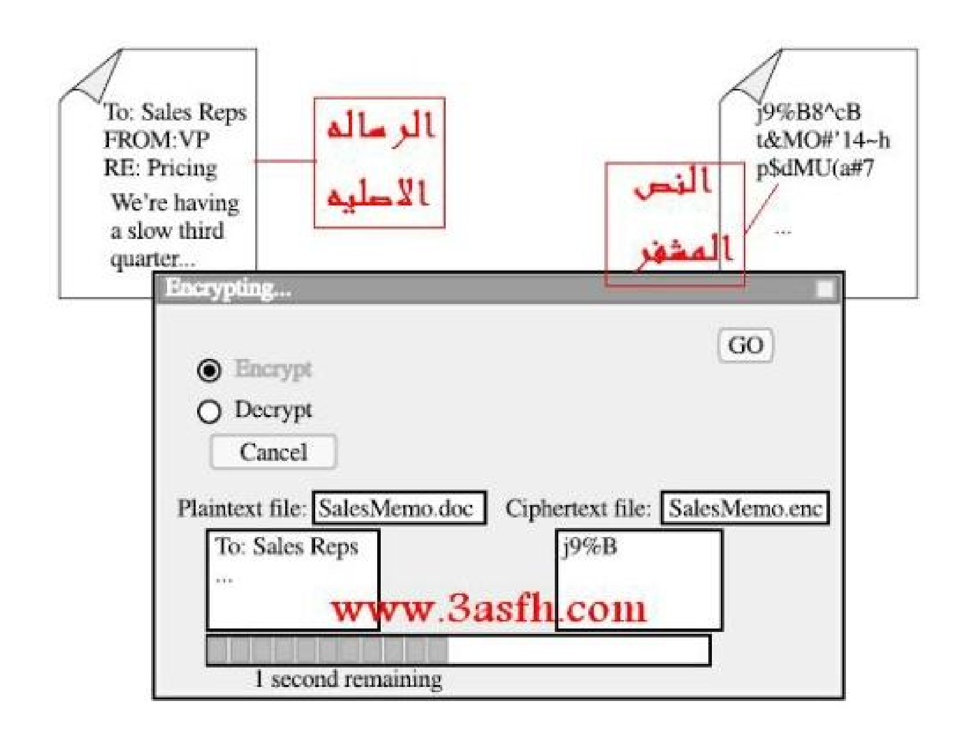
ولنضرب مثال بسيط يوضح العملية :

الأخ محمد هو مندوب مبيعات في احد الشركات الكبرى ، وقد وصلته رسالة من مدير قسم المبيعات الأخ علي ، هذه الرسالة تحتوي على معلومات سريه جدا خاصة بالأسعار الجديدة للمنتجات وبعض الأمور الخاصة في الشركة .

الآن قرر أخونا محمد هو والمدير علي الحفاظ على هذه الرسالة السرية لديهما فقط ، فما هو السبيل لذلك ؟ قد يستطيع الأخ محمد حفظ الرسالة لديه في المكتب في الدرج (لكنه يخاف من أن تسرق من الدرج) ، أو ربما يحفظ تلك الرسالة في رأسه (لكن للأسف الرسالة طويلة جدا) ، و هو يحتاج إليها في عمليه البيع لان بها أسعار المنتجات ... (لذلك يجب أن يحملها معه) .

قد يقوم الأخ محمد بحفظ هذه الرسالة في جهازه المحمول والقيام بوضع باسورد على النظام وبعض الصلاحيات ، لكن رأينا قبل قليل أن هذه الطريقه غير كافيه ، قد يضيع الجهاز المحمول أو قد يسرق منه وبعدها تنكشف كل المعلومات وتضيع الشركة .

أخيرا قرر أخونا محمد تشفير تلك الرسالة ، وبما انه لا يعرف أي شيئا في البرمجة قام بشراء برنامج من الأخ romansy يقوم بتشفير الرسالة ، هذا البرنامج بسيط في عمله -User friendly ويحتوي على ثلاثة أزرار واحد للتشفير، و واحد لفك التشفير ، و واحد لتحميل الملف من الجهاز ، وبكل بساطه لكي يعمل يقوم محمد بتحميل الملف (الرسالة) ، والضغط على زر التشفير Encryption وبعدها تتحول الرسالة إلى كلام غير مفهوم (مشفر) ، وفي حال حصل المخترق على تلك الرسالة الناتجه فبالتأكيد لن يفهم شيء..



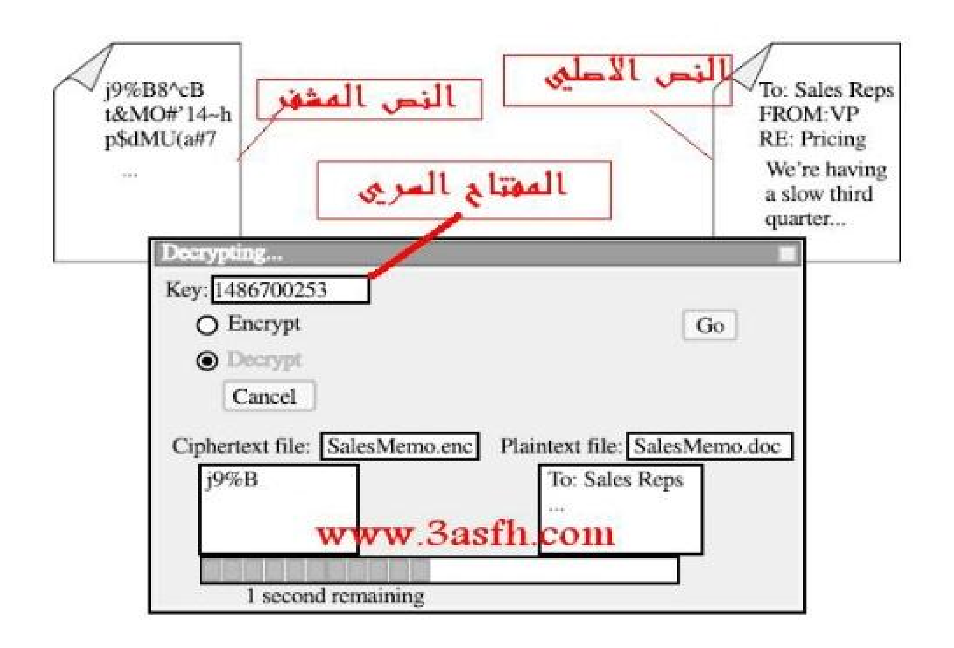
الآن اذا أراد أخونا محمد فك التشفير (العملية العكسية) كل ما عليه هو تحميل الرسالة المشفرة وبعدها الضغط على زر فك التشفير ، وترجع الرسالة إلى حالتها الاصليه..

المشكلة الحقيقة هنا ، في حال حصل المخترق على برنامج التشفير هذا الخاص بالأخ Romansy هنا كل ما على المخترق تحميل الملف المشفر وضغط زر فك التشفير وبعدها يحصل على الرسالة !! من الممكن أن تقول "كيف يستطيع الحصول على برنامج التشفير" ، الجواب بالتأكيد هو سيحصل عليه ، اذا كنت تستطيع إخفاء هذا البرنامج فلماذا لا تخفى الرسالة من الأصل ، ولا تحتاج لبرنامج لبرنامج الأخ Romansy ولا أصلا للتشفير ككل .

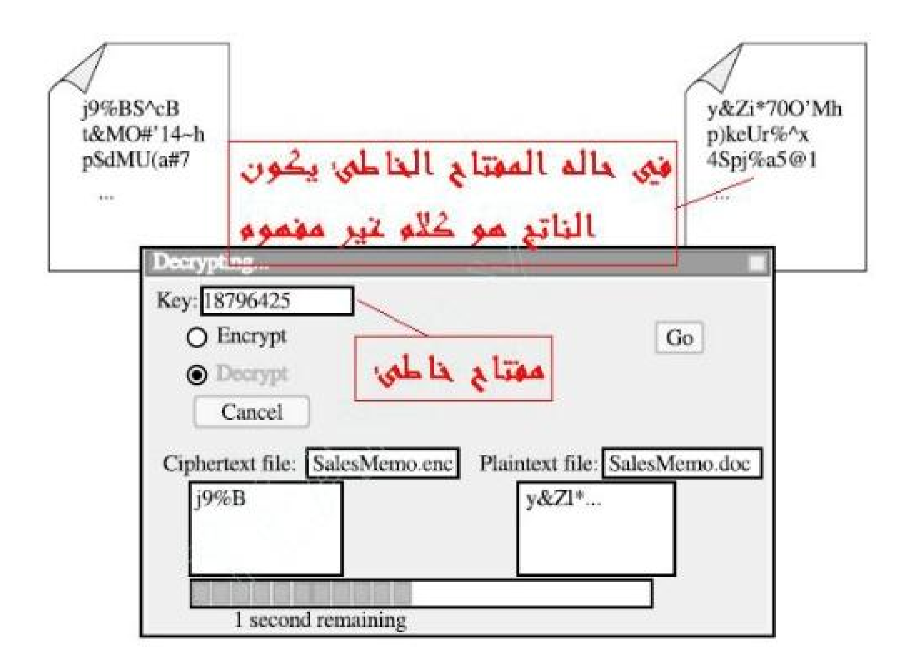
بالتأكيد أخونا محمد لا يستطيع إخفاء هذا الملف ، اذا ما العمل اذا ؟؟ مممم بالتأكيد هو يحتاج إلى شيء إضافي ، ألا وهو المفتاح السري Secret Key .

قام الأخ محمد بإبلاغ romansy وقد تم اضافه خاصية إدخال المفتاح ، الآن البرنامج لكي يعمل يجب أن يقوم الأخ محمد بتحميل الملف (الرسالة) إلى البرنامج ، ويدخل المفتاح سري (إي رقم يحفظه تماما) بعدها يضغط على زر التشفير ، والناتج هو الملف المشفر (الغير مفهوم).

في حال أراد الأخ محمد بالقيام بالعملية العكسية ، كل ما عليه تحميل الملف المشفر إلى البرنامج ، وإدخال الرقم السري الذي استخدمه للتشفير ، والضغط على زر فك التشفير ، والناتج هو الملف الأصلي (أو الرسالة) .



الآن في حال أخونا المخترق حصل على البرنامج والرسالة المشفرة وأراد القيام بالعملية العكسية ، هو للأسف لا يملك المفتاح السري ، وسوف يكتب ويخمن ووو وفي كل مره يكتب مفتاح سري خاطئ سوف يكون هناك ناتج غير مفهوم .



وهكذا ، أي احد لن يستطيع الحصول على الرسالة الاصليه إلا بعد كتابه المفتاح السري ، وإذا ادخل المفتاح الخاطئ سوف يكون الناتج غير مفهوم ،وحتى لو كان هذا المفتاح الخاطئ اقل من المفتاح السري برقم واحد .

طريقه التشفير السابقة التي استخدمناها تسمى (**التشفير بالمفتاح المتناظر أو المتماثل Symmetric key Cryptography**) والبعض يسميها **Secret key Cryptography** ، أيضا اسم (**التشفير التقليدي Conventional Encryption**) ، لكن نحن نأخذ المصطلح الأول .

بالنسبة إلى علم التشفير هناك الكثير من المصطلحات لها نفس المعنى ، على العموم سأحاول وضع الأغلب والأشهر هنا .

في حال أردت أن تحول المعلومات المفهومة إلى غير مفهومه تسمى العملية **تشفير Encryption**

في حال أردت أن تحول المعلومات الغير مفهومه إلى مفهومه تسمى العملية **فك التشفير Decryption**

للتشفير أو فك التشفير ، يجب أن تتبع **خوارزمية Algorithm** معينه ، الخوارزمية هي مجموعه خطوات مرتبه بطريقه معينة تؤدي هدف معين ، بالطبع مفهوم الخوارزمية مفهوم لدى اغلب المبرمجين ، وتستطيع تطبيق الخوارزمية بأي لغة برمجه ، المهم في التشفير الخوارزمية ممكن أن تكون علميه رياضيه معقده للغاية وممكن أن تكون عمليه جمع بتات أو عمليه XOR (في اغلب الخوارزميات التعامل بيكون على حسب البت bit ، لذلك سوف تستخدم عمليات الـ Bitwise operation) .

البيانات (أو الرسالة) التي نريد إجراء عمليه التشفير عليها تسمى (**النص الواضح plain text أو النص الأصلي clear text**).
البيانات بعد التشفير تسمى (**النص المشفر cipher text**) .

الخوارزمية هي التي تقوم بالتشفير ، وتحتاج إلى **مفتاح Key** ، قد يكون رقم أو مكون من عده أرقام.

لدينا أيضا صديقنا **المخترق attacker** ، وهو الذي نخاف منه ، ونشفر حتى لا يطلع عليها ، وهدفه هو التخريب أو السرقة أو التلاعب بمشاعرنا ☺ .

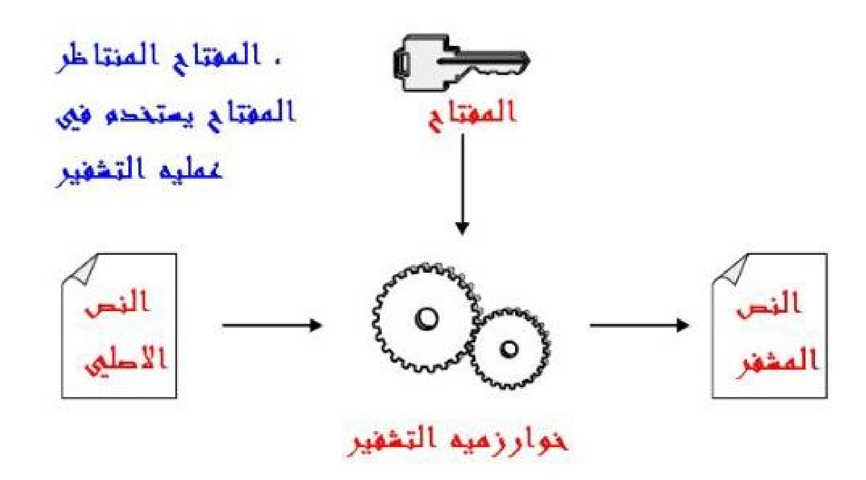
العلم الذي يستخدم لكسر الخوارزميات وإيجاد نقط الضعف بها يسمى **Cryptanalysis** والشخص الذي يقوم بهذا العلم يسمى **cryptanalyst** (يكمن تسميته كاسر الشفره) .

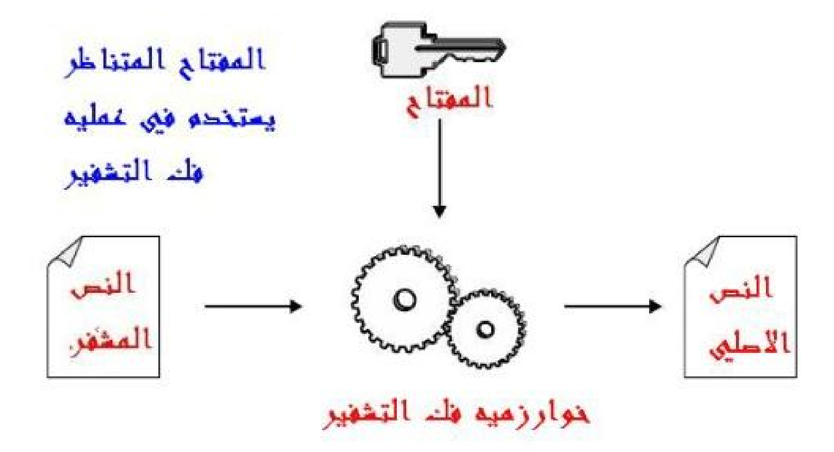
كل الخوارزميات يمكن أن تكسر وتخترق ، لكن الشيء الجيد هو انه اذا كانت الخوارزمية قويه ، قد يأخذ وقت الكسر وقتا طويلا جدا ، عندها تكون النتيجة too late . لذلك الـ cryptanalyst يقوموا بإيجاد نقاط الضعف التي تساعد أي شخص باختراق تلك الخوارزميات في أسرع وقت !! والمخترق بالتأكيد يقوم باستخدام النتائج الخاصة بالـ cryptanalyst وأيضا لديه العديد من الطرق والأدوات الخاصة.

اذا النتيجة هي أن الـ cryptographer هو الذي يطور انظمه التشفير ، الـ cryptanalyst يقوم بإيجاد نقاط الضعف فيها وهي خدمه عظيمة للمشفرين ، حيث أنهم يكتشفوا الثغرات ويقوموا بنشرها للجميع ، بعكس المخترق الذي يكشف الثغرة ويحتفظ بها لنفسه لأجل الاختراق.

ما هو المفتاح ، وما هي أهميته

في التشفير بالمفتاح المتناظر Symmetric المفتاح الذي يستخدم للتشفير هو نفسه الذي يستخدم لفك التشفير ، ومن هنا جاءت كلمه المتناظر(أو المتماثل) ، يعني هو نفسه من الطرفين ، الصوره التالية توضح العملية :





لاحظ انه في حاله شفرت الرسالة بخوارزمية معينه (مثلا DES) ومفتاح معين (مثلا 10) ، الآن في حاله فك التشفير يجب أن استخدم نفس الخوارزمية ونفس المفتاح والا فلن احصل على النص الأصلي.

نعود للسؤال ، ما هي أهميه المفتاح ؟؟؟

لقد رأينا في المثال السابق (محمد وعلي ، عندما لم يستخدموا مفتاح سري) انه في حاله حصول المخترق على البرنامج (أو معرفه الخوارزمية المستخدمة في التشفير) فانه بكل بساطه يستطيع إرجاع النص المشفر إلى النص الأصلي .

قد يسأل أحدكم سؤال "**حسنا ، لماذا لا اخترع خوارزمية وأبقيها سريه عن الجميع وبهذا لا يعرفها المخترق وبالتالي لا احتاج إلى مفتاح**" ؟؟

سؤال جيد ، ولكن له عده مشاكل ، أولا لان المخترقين دائما يكسروا ويخترقوا الخوارزمية (سنشاهد بعد قليل بعضا من الامثله الواقعية) ، ثانيا ، في حال انك لم تكن خبير في التشفير ولا تستطيع تطوير خوارزمية خاصة بك (مثل أخونا محمد) يجب في تلك الحالة أن تثق بالشركة المنتجة للبرنامج (الخوارزمية) الذي تستخدمه (في هذه الحالة يجب أن تثق ب Romansy) ؟
هل يستطيع أحدكم أن يمنح شركه ما كل هذه الثقة ، بالطبع لا .
وهنا يأتي السؤال الحقيقي ، من تثق بحفظ أسرارك ، **خوارزميه يجب أن تكون سريه من الجميع**، أم **الخوارزمية التي تؤدي عملها بشكل جيد وحتى لو عرفها الجميع** ، وهنا يأتي دور المفتاح في حال اخترت الخيار الثاني .

المفتاح يجعلك تشعر بالارتياح التام ، لأنك اذا شفرت الخوارزمية باستخدام المفتاح ، سوف تكون مهمتك الحفاظ على المفتاح فقط ، بالتأكيد هو أسهل بكثير من الحفاظ على الخوارزمية التي اخترعتها.

أيضا في حال استخدمت مفتاح تشفير لكل رسالة ، في حال تم كسر احد المفاتيح ، فان باقي الرسائل تكون سريه وغير مكشوفة ، على العكس اذا استخدمت خوارزمية من تطويرك وتم كشفها فان كل الرسائل سوف تنكشف أيضا.

السؤال السابق نطرحه مره أخرى ،" **في حال شفرت الرسالة بخوارزمية لا يعرفها احد ، وإذا استخدمت مفتاح فلن ابلغ احد بطول المفتاح** " هل تكون الرسالة أمنه ؟

و هناك ثلاثة أجوبه :

الجواب الأول : هم دائما يعرفون الخوارزمية:

المخترقين سوف يعرفوا الخوارزمية مهما فعلت ، ولا أي واحد في تاريخ التشفير تمكن من إبقاء خوارزميته سريه ، لطالما تمكن الجواسيس في الحرب كشف الخوارزميات سواء باستخدام عمليات رياضيه أو أجهزه لكسر الشفرات ، أو حتى يوظفوا جواسيس لدى العدو ، أو يسرقوا الخوارزمية ، أو يسرقوا الجهاز المستخدم للتشفير .

في الحرب العالمية الثانية ، تمكن الجنود البولنديين من سرقه الجهاز الألماني الذي كان الألمان يستخدموه للتشفير (اسمه **Engima**) وتم بيعه للبريطانيين (الحلفاء) وبعدها تمكن هؤلاء الحلفاء من كسر اغلب الرسائل الالمانيه !!

وبدون أي سرقه ، فان الـ cryptanalysts يستطيعوا ببساطه كسر الشفرات ، ففي الحرب العالمية الثانية تمكن كاسري الشفرات الأمريكيين (اسمهم **Code breaker**) من معرفه طريقه عمل الجهاز المستخدم في التشفير لدى اليابانيين(بدون الحاجة إلى سرقه الجهاز) .

مثال آخر ، هناك خوارزمية اسمها RC4 اخترعت من قبل شركه RSA في عام 1987 لكن لم تنشر ، كل الـ cryptanalysts والمشفرين اجمعوا في ذلك الوقت أن هذه الخوارزمية آمنه جدا وتجعل البيانات سريه للغاية ، ولم تنشر تلك الخوارزمية لأغراض بيع برامج للتشفير (وليس لأغراض عسكريه) ، المهم في 1994 قام احد الهكرز بوضع الخوارزمية مشروحة بالتفصيل في الانترنت ! كيف عرف هذا الهكرز الخوارزمية ؟؟ بالتأكيد من خلال برامج Disassembly And Debugger، وهي برامج تستخدم لفتح الملفات التنفيذية وتتبعها سطر بسطر وتغيير الأكواد والكثير من الأمور التي يعرفها الكراكرز . حاليا خوارزمية RC4 تستخدم كجزء من بروتوكول الـ SSL وهي من احد الخوارزميات التي تستخدم المفتاح المتناظر للتشفير.

بعض الأحيان ممكن أن تبقى الخوارزمية سريه لبعض الوقت ولكنها تنكشف في النهاية ، مثلا في الحرب العالمية الثانية استخدم الأمريكان لغة Navajo وطبعا اليابانيين لم يكونوا على علم بهذه اللغة ، لذلك الرسائل الامريكيه كانت مشفره . لكن حاليا اغلب الجيوش تحتوي على فريق كامل من العلماء باستطاعتهم تعلم أي لغة مهما كانت وبأسرع وقت .

الجواب الثاني : لا تستطيع جنى المال من الخوارزمية السرية:

لأنك في حال عملت برنامج وقمت ببيعه فبالتأكيد سوف يقوم احد الهاكر باستخدام طرق الهندسة العكسية والوصول إلى خوارزميتك (كما حصل في RC4) . لذلك الخوارزمية التي طورتها سوف تستخدمها لنفسك أنت وحبيبتك فقط ☺ .

الجواب الثالث : الخوارزميات المعروفة هي أكثر آمانا:

لأنها تكون مثبته أنها آمنه من قبل جميع الـ cryptanalysts والمشفرين ، من الممكن أن هؤلاء

cryptanalysts قد اقروا بعدم وجود ثغرات بخوارزمية معينه ، بعدها بفترة قمت أنت بكشف ثغره فيها ، نعم ممكن ولكن احتمال ضعيف جدا جدا .

في حال اقر الجميع بان الخوارزمية آمنه ولا توجد ثغرات بها ، فان فرصه إيجاد ثغره ضئيلة للغاية .

توليد المفتاح *Key Generation* :

في التشفير المتناظر ، المفتاح عبارة عن رقم وطوله على حسب نوع الخوارزمية (64 بت مثلا)، ويمكن توليده بصوره عشوائية ، السؤال هنا كيف يتم توليد الأرقام العشوائية ؟

حسنا ، الأرقام العشوائية هي بكل بساطه أرقام مثل (3،1،5،100) يتم اختيارها بشكل عشوائي، اغلب المبرمجين يعرفون قيمه هذه الأعداد فهي تستخدم بكثرة في عده نواحي مثل الألعاب Game، نمذجه ومحاكاة الحاسب Simulation And Modeling والتشفير Cryptography وغيرها من المجالات .

في التشفير ، أهم ما يجب أن يتوفر في هذه الأعداد هو أن لا تتكرر أبدا ، أيضا أن تتجاوز الاختبارات الاحصائيه ، الاختبارات الاحصائيه هي مجموعه اختبارات يتم تطبيقها على الأعداد (أو العدد) لكي تعرف هل هي عشوائية أم لا..

لنفترض لدينا مجموعه من الأعداد (ألف عدد مثلا) ، وقمنا بسؤال احد الذين يقومون بهذه الاختبارات "هل هذه الأعداد عشوائية أم لا "، كل ما يقومه هذا الشخص (الإحصائي) بالقيام بتحويل الأعداد أولا إلى الترميز الثنائي Binary Format أي يقوم بتحويل الأعداد إلى 0 و1 ، بعدها يقوم بإجراء عده اختبارات على هذه الأعداد ، الاختبارات تكون عبارة عن عده اسئله:
هل العدد 1 يظهر بنفس تكرار 0 ؟ أم أكثر أم اقل ؟
هل العددين 1و 0 يظهران بشكل محدد كل مره ؟ (مثلا تأتي 1 أولا بعدها 0) ؟
وغيرها من الاسئله....



المهم ، بعدها في حال نجح الاختبار ، تكون الأعداد العشوائية التي أعطيتها للإحصائي محتمله أن تكون عشوائية !! لا نستطيع أن نقول هي عشوائية بصوره مؤكده 100% ، لماذا ، سنعرف لاحقا .

مولدات الأرقام العشوائية A Random Number Generator اختصارا RNG :

عرفنا أن الأعداد العشوائية هي أعداد تكون بصوره عشوائية وغير مرتبه ، حسنا السؤال الذي يطرح نفسه : من أين يمكن الحصول على هذه الأعداد العشوائية ؟؟

هناك مصدرين:
الأول هو لتوليد أعداد عشوائية حقيقية أو كاملة RNG أو True RNG ، وهنا في هذه الحالة سوف نستخدم أجهزه خاصة لتوليد الأعداد (تأخذ مدخل يتغير باستمرار) مثل قياس الظروف الجوية ، حساب سريان التيار الكهربائي وغيرها وهذه الأعداد بالطبع سوف تجتاز الاختبار الإحصائي..

وإذا طلبنا من هذه الاجهزه أعداد عشوائية أخرى ، فلن نحصل على نفس الناتج أبدا ، ولذلك لان المخرج (الأعداد) تعتمد على مدخل غير ثابت (يتغير باستمرار) ، لذلك فإن الأعداد العشوائية الناتجه من هذه الأجهزة لا تتكرر ابداً ، ولهذه تسمى بالأعداد العشوائية الصحيحة أو الكاملة True Random Number .

قرأت عن أن شركه Intel تقوم باستخدام RNG يوضع داخل النظام ويقوم بحساب الحرارة أو شيء مشابه ، وهو مصدر يتغير باستمرار ، أيضا هذا الجهاز لا يأتي مع أي معالج بنتيوم (إلا مع الطلب) ، لكن يحتمل ذلك في السنوات المقبلة .

شركات أخرى مثل nCipher, Chrysalis : تبيع أجهزه تسمى cryptographic accelerators هذه الاجهزه تأتي بـ RNG ، (سوف نلقي نظره بسيطة على cryptographic accelerators بعد قليل) .

المصدر الآخر لتوليد الأرقام العشوائية هو Pseudo-Random Number Generator (الأعداد العشوائية المزيفة) :

من أين يمكننا الحصول على أعداد عشوائية اذا كنا لا نملك هذه الاجهزه ، الجواب باستخدام مولد الأعداد المزيفة ، وهو عبارة عن خوارزمية لتوليد هذه الأعداد "المزيفة" ، بالتأكيد كلمه "مزيفه" يسبب لك بعضا من الحيرة ، لكنه سيتضح بعد قليل .

هنا في حاله الأعداد العشوائية المزيفة اذا استخدمنا الخوارزمية وولدنا الأعداد (مثلا ألف عدد) ، بعدها ذهبنا إلى صديقنا الإحصائي وقمنا باختبار هذه الأعداد ، الناتج هو أن هذه الأعداد سوف تنجح أيضا في الاختبار (مثلها مثل T-RNG) ، لكنها يحتمل أن تكون عشوائية .

الذي يجعل هذه الإعداد مزيفه هو أنها تتكرر (وألف خط تحتها) اذا شغلت مولد الأعداد المزيفة في جهازين مختلفين سوف يطلع بنفس الناتج ، اذا شغلت البرنامج بعد سنه سوف يطلع بنفس الناتج .

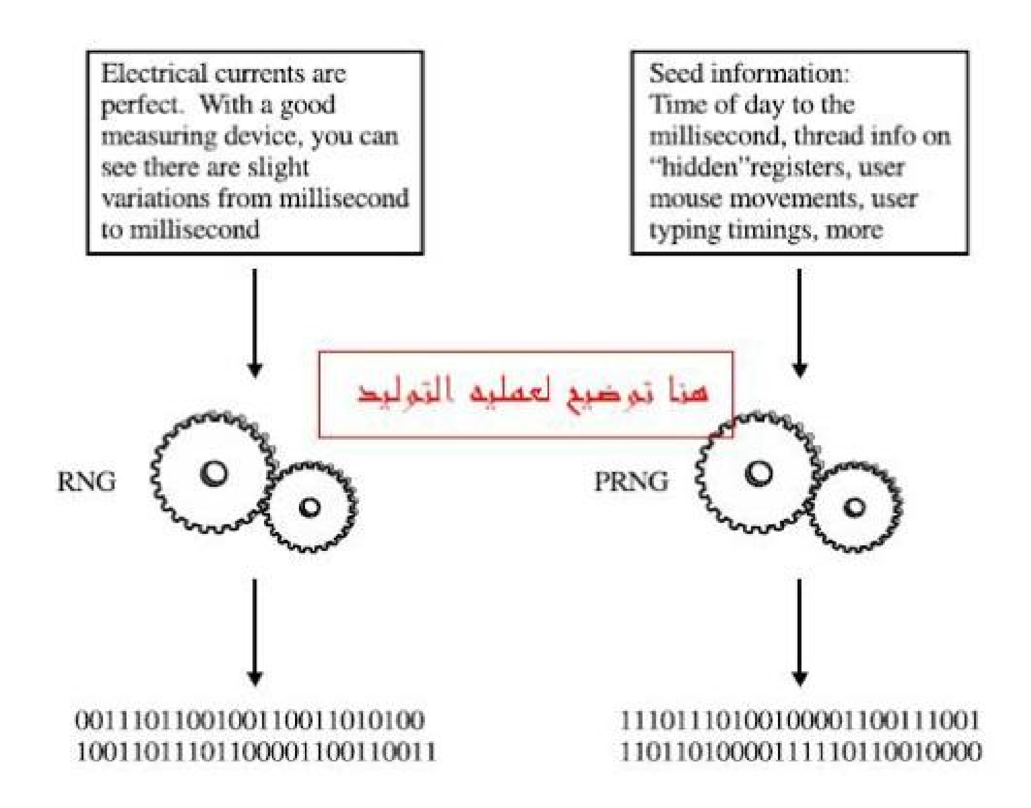
لذلك قبل قليل قلنا أن النتيجة التي يخرج بها الإحصائي هي محتمله أن تكون عشوائية وليس عشوائية 100% .
اذا الإحصائي يعطينا أجابه على أن الأعداد عشوائية فقط ، ولكن هو لا يعرف هل هي تتكرر أم لا ، يعطينا فقط نصف الاجابه.

حسنا ، اذا كانت الأعداد في PRNG تتكرر ، اذا ما هو الشيء الجيد فيها ؟؟

لأنه بكل بساطه يمكنك أن تغير الناتج عن طريق ما يعرف بالـ Seed (البعض يترجموها بالبذرة) ، كما هو الحال مع RNG تأخذ المدخل من (قياس الظروف الجوية، التيار) فإن PRNGتأخذ المدخل من (البذرة Seed). اذا غيرت المدخل Seed سوف تتغير المخرجات ، في الـ RNG المدخلات تتغير بنفسها بدون الحاجة إليك ، في PRNG يجب عليك أن تغير المدخل Seed في كل مره أردت فيها الحصول على عدد عشوائي .

ما هي البذرة أو الـ Seed ؟؟

هذه البذرة ممكن أن تكون مثلا الوقت (كل جزء من الألف من الثانية millisecond) ، أو مثلا مدخل من الكيبورد ، أو حركه من الماوس ، أو موقع الماوس في الشاشة (عدد البكسلات من الجهة الافقيه والعاموديه) ، باختصار يعني لا يمكن توقع قيمتها .



قد تتساءل الآن : "لماذا لا استخدم الـ Seed فقط بدون الـ PRNG " ؟

لسببين ، الأول هو الحاجة للسرعة ، فجمع الـ Seed يأخذ وقت طويل نوعا ما ، مثلا أنت بحاجه إلى عده الآلاف عدد ، اذا استخدمنا الـ Seed فانه يأخذ وقتا حتى يجمع الأعداد المطلوبة !! تخيل مثلا جمع الSeed كان عبارة عن حركه للماوس (واخذ الموقع) ، هنا لكي نولد 1000 عدد فإننا بحاجه إلى 1000 حركه ، مشوار طويل واضاعه للوقت .

ولكننا اذا استخدمنا الـ PRNG كل ما علينا هو إدخال Seed واحد (بطول 160 بت تقريبا) وبعدها نبدأ بتوليد الأعداد باستخدام هذا ال Seed، وسوف تخرج لنا الآلاف من الأعداد في غضون ثواني .

السبب الآخر هو الـ (entropy) ، والـ PRNG يزيد من قيمه هذا المتغير ، مما يجعل النتيجة عشوائية أكثر .

أيضا اغلب الـ PRNG تستخدم مفهوم يسمى **message digests** ، على العموم فكرته بسيطة مثل الخلاط blenders ، الخلاط العادي الذي يستخدم في المطابخ يأخذ العديد من الاطعمه ويخلطها ويخرج الناتج خليط من الاطعمه التي دخلت ، و الـ message digests تأخذ البتات وتدخلهم هذا الخلاط ويخرج الناتج خليط بشكل عشوائي .

سوف ندرس خوارزميات توليد الأعداد العشوائية المزيفة (في النسخة النهائية من الكتاب)، ونقوم بتطبيقها مثل LCG ، بالمناسبة الدوال rand , random تولد أعداد مزيفه ، و على حسب ما قرأت انه random تولد أعداد أكثر عشوائية .

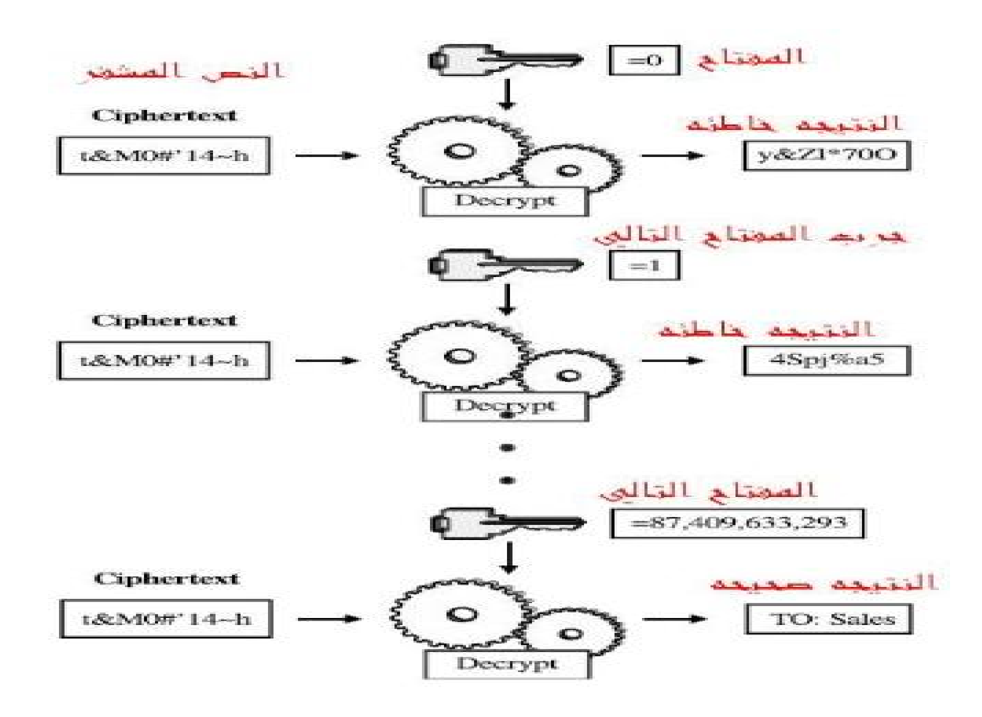
الهجوم على البيانات المشفرة *Attacks on Encrypted Data*

المخترق هو الشخص الذي يريد سرقه المعلومات ، ولكي يحصل على المعلومات يجب أن يفك تشفير البيانات المشفرة ، ولفك التشفير فإن لديه طريقتان ، معرفه المفتاح ، أو كسر الخوارزمية .

الهجوم على المفتاح Attacks on Keys :

هنا يقوم المخترق بتطبيق هجوم يسمى القوه العنيفة **brute-force attack** وطريقته هو أن يجرب مفتاح مفتاح إلى أن يصل إلى المفتاح المطلوب.

لنفرض أن المفتاح يقع في النطاق من 0 إلى 100,000,000,000 (مائه بليون) ، أولا يقوم المخترق بإدخال الرقم 0 إلى خوارزمية فك التشفير ويقوم بإدخال الرسالة المشفرة أيضا ، بعدها ينظر إلى النتيجة هل هي معقولة أو مفهومه ، اذا كانت الاجابه نعم فإذاً المفتاح هو 0 ، أما اذا كانت الاجابه بـ لا ، فيقوم بتجربة المفتاح 1 و بعدها 2 و100 و ... 100,000,000,000.



تذكر أن الخوارزمية تقوم بعملها بغض النظر هل المفتاح صحيح أم خاطئ ، وفي كل مفتاح خاطئ النتيجة تكون غير مفهومه ، لذلك يجب على المخترق النظر إلى النتيجة في كل مره يجرب فيها مفتاح ، المخترق الذكي يكتب برنامج يقوم بفحص الرسالة هل هي مفهومه (ضمن نطاق الأعداد الابجديه) ؟ نعم اذا أرسل هذا هو المفتاح ، والا جرب الرقم التالي ...

في حاله 100 بليون رقم ، قد يحصل المخترق على المفتاح من أول تجربه ، ويمكن من التجربة الاخيره ، على العموم في المتوسط قد يحصل عليه في 50 بليون محاوله ، السؤال هو كم من الوقت تأخذ عمليه تجربه 50 بليون مفتاح ؟ 3 سنوات ؟ 3 شهور؟ 3 أيام ؟

لنفرض انك تريد الحفاظ على الرسالة لمده 3 سنوات ، وان المخترق يستطيع محاوله 50 بليون مفتاح في 3 دقائق اذا عليك تكبير المفتاح وجعل المدى من 0 إلى 100 بليون بليون بليون بليون بليون بليون ، وهكذا يأخذ المخترق وقتا أطول حتى يجد المفتاح الصحيح..

هذا المفهوم يعرف بـ **حجم المفتاح Key Size** ، النقود تقاس بقيمتها ، الذهب بالوزن ، المفتاح في التشفير يقاس بعدد البتات bits ، والمفاتيح في التشفير المتناظر تكون عاده بطول 40 bit أو 56 bit أو 128 bit (وهو الأفضل) . وهكذا ...

مدى الـ 40 بت هو من 0 إلى 1 تر ليون
مدى الـ 56 بت هو من 0 إلى 72 كر وليون quadrillion
مدى الـ 128 بت هو من 0 إلى ..... أفضل أن نقول 128 بت فقط

باختصار ، اذا كنت تريد أن تجعل عمل المخترق أكثر صعوبة ، قم بتكبير حجم المفتاح ، المفتاح الكبير يضمن لك حماية اكبر .

في عام 1997 ، تم كسر الـ 40 بت في ثلاث ساعات ، و تم كسر 48 بت في 280 ساعة .. (كل بت زيادة يأخذ ضعف المدة السابقة ، مثلا 40 بت تأخذ 3 ساعات ، 41 بت تأخذ 6 ساعات ،وهكذا) .

في 1999 ، تم كسر 56 بت في 24 ساعة (تم كسره من قبل Electronic Frontier Foundation وتم استخدام جهاز خاص فقط لكسر خورزميه DES) .

بالطبع المخترق لن تكون لديه هذه الإمكانيات وبالتالي سوف يأخذ وقتا أطول بكثير من الشركة السابقة ، ولكن نحن هنا نفترض أن المخترق عبارة عن حكومة أو الـ FPI نفسها .

حاليا 128 بت هو أفضل حجم للمفتاح المتناظر ، ربما في حال كسره سوف ننتقل إلى 512 بت (وهو اكبر حجم للمفتاح ونظريا لا يمكن كسره بهذه الطريقه (brute-force attack) ابدآ) .

ومن الممكن أن يقوم المخترق بالهجوم على المفتاح ولكن بطريقه أخرى ، وهي محاوله توليد PRNG بنفس البذرة Seed التي قمت بإدخالها في الـ PRNG ، اذا استخدمت Seed صغير ، قد يقوم المخترق بتجربة رقم رقم إلى أن يجد ال Seed الصحيح (قد حصل هذا لشركه نت سكيب ، سأذكره بعد قليل) ، لذلك عليك عندما تختار Seed أن تختار واحد ممتاز ولا يمكن تخمينه ...

نظره حول Netscape's Seed :

أن التشفير بالمفتاح المتناظر هو احد مكونات SSL الذي اخترع من قبل العلماء في شركه نت سكيب (احد العلماء بل رئيس العلماء في فتره التسعينات هو العالم العربي المصري **طاهر الجمل** ، والذي اخترع الخوارزمية التي تعرف بـ ELGAMAL نسبه إلى اسمه ، سأتكلم عنه في النسخة النهائية) ، وفي وقت توليد الاتصال في SSL يجب أن يولد رقم عشوائي ، وقد استخدمت الشركة PRNG يجمع المعلومات (الوقت + رقم العملية process ID) واستخدم كبذره Seed للمولد PRNG .

بالنسبة إلى process ID فيمكن الحصول عليه من خلال الدخول إلى نفس الجهاز الذي ولده ، أو يمكن تطبيق هجوم brute-force attack حيث طوله فقط 15 بت ، أما بالنسبة إلى الوقت ، استخدمت الشركة الثواني (وليس جزء من الألف من الثواني) ولذلك هناك 60 ثانيه فقط .

على العموم في 1995 قام Goldberg و Wagner بإيجاد ال Seed وبالتالي إيجاد المفتاح في اقل من دقيقه ، سواء كان المفتاح 40 بت أو 128 بت ، سوف يأخذ اقل من دقيقه !!

وقد قامت نت سكيب بعدها باضافه Seed جديد يعتمد على عده عوامل:
. mouse position, memory status, last key pressed, audio volume, and many others
وهكذا يصعب إيجاد ال Seed .

كسر الخوارزمية Breaking the Algorithm :

الطريقه الأخرى للهجوم على البيانات المشفرة هي كسر الخوارزمية ، وهنا يعتمد على قوه ملاحظه وذكاء المخترق ، مثلا لاحظ أن رقم معين يظهر في مكان معين ، هنا يستطيع تخمين هذا العدد بعد تجربه العديد من المحاولات حتى يحصل على النص الأصلي .



مثل تلك الخوارزميات الضعيفة يمكن أن تخترق وحتى لو كان حجم المفتاح كبير .

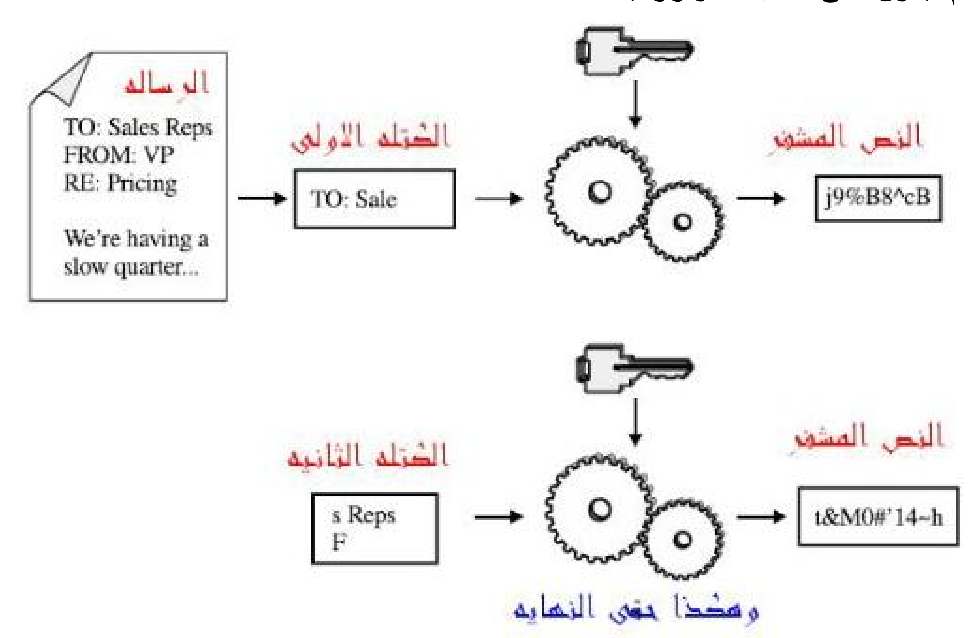
بعد أن عرفنا ما هو التشفير بالمفتاح المتناظر وأهميه المفتاح ، تبقى أن نعرف ما هي الخوارزميات التي تنطوي تحت هذا النوع من أنواع التشفير (تذكر أن هناك نوعين ، الأول هو التشفير بالمفتاح المتناظر ، والثاني بالمفتاح غير المتناظر) .

هناك نوعين من أنواع التشفير بالمفتاح المتناظر Symmetric Key Cryptography وهما :

*شفرات الكتل Block Cipher
*شفرات التدفق Stream Cipher

شفرات الكتل *Block Cipher*

شفرات الكتل (كما هو واضح من اسمها) تعمل على Block (كتله) من البيانات في كل مره ، عندما تزود الخوارزمية (أو برنامج التشفير) بالنص الأصلي هنا في هذا النوع يقوم بتقسيم النص إلى عده من الكتل ، كل كتله حجمها يكون 64 بت وأحيانا 128 بت (16 بايت) ، طبعا الحجم بيكون على حسب الخوارزمية المستخدمة



نأخذ مثال بسيط :

لنفرض أن طول النص الأصلي 227 بايت ، والخوارزمية تأخذ 16 بايت في كل مره ، الآن في مرحله التشفير ، ندخل المفتاح ، والكتلة الأولى (أول 16 بايت) ، ونبدأ عمليه التشفير ، والناتج هو نص مشفر بطول 16 بايت أيضا . بعدها نأخذ الكتلة الثانية (ثاني 16 بايت) ، و نبدأ في عمليه التشفير ، والناتج هو أيضا نص مشفر بطول 16 بايت ، وهكذا ستستمر العملية 14 مره (وهنا تكون شفرت 224 بايت من النص الأصلي).

الآن بقيت 3 بايتات ولكن الخوارزمية لا تعمل إلا اذا كانت الكتلة بطول 16 بايت ، اذا ما العمل ؟؟ هنا سوف تستخدم مفهوم جديد اسمه **الحشو Padding** وهو بكل بساطه يعمل على اضافه بايتات اضافيه إلى الكتلة الناقصة حتى تكتمل وتصبح بالحجم المطلوب. وهناك عده طرق للحشو

سوف نذكر اشهرها ، على العموم في حال استخدمنا أي طريقه يجب أن تكون مفهومه للشخص الذي نريده أن يفك تشفير الرسالة أي يعرف هذه البايتات هي بايتات اضافيه.

الطريقه الأشهر للحشو : هي أن نعرف أولا عدد البايتات التي سوف نضيفها إلى الكتلة الناقصة ، في المثال السابق كانت (13 بايت) ، بعدها نقوم بتكرار هذا العدد في كل بايت في الكتلة ، أي انه سنحشو العدد "13" ثلاثة عشر مره ، وتسمى هذه الطريقه **PKC#5**

في عمليه فك التشفير ، سننظر إلى الكتلة الاخيره ونرى إن كان هناك عدد ما يتكرر في كل خانه من خانات الكتلة الاخيره ، ومنه سوف نعرف هل كان هناك حشو أو لا .

طبعا عند التشفير ، في حاله الكتلة الاخيره تساوي 16 بايت فلا يوجد داعي للحشو (أصلا لا يوجد مكان للحشو).



المشكلة في هذا النوع "شفره الكتل " Block Cipher هو انه في حاله كانت هناك كلمات تتكرر كثيرا في النص الأصلي ، فإنها بعد التشفير سوف يكون النص المشفر متشابه أيضا .

مثلا الاسم " wajdy essam " ظهر 3 مرات في مواقع مختلفة في النص الأصلي (أو الرسالة) . مثلا ظهر في الكتلة الأولى (أول 16 بايت) والكتلة الرابعة (رابع 16 بايت) والكتلة العاشر(عاشر 16 بايت).

الآن عند التشفير ،
دخلت الكتلة الأولى "Wajdy Essam" وتتشفر إلى "Selrurjgqm"
دخلت الكتلة الرابعة "Wajdy Essam" وتتشفر إلى " Selrurjgqm "
دخلت الكتلة العاشرة "Wajdy Essam" وتتشفر إلى " Selrurjgqm "

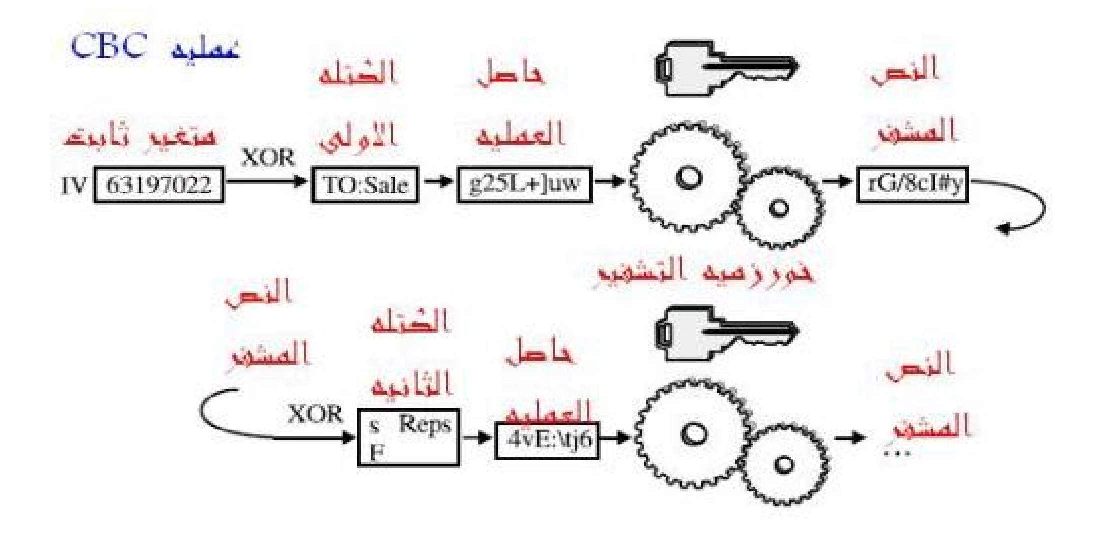
نلاحظ انه يمكن للمخترق معرفه أن الثلاثة " Selrurjgqm " هي عبارة عن جمله واحده ، وهنا ممكن أن يكسرها بسهوله .

الحل لتجنب مثل هذا التكرار هو استخدام أساليب تسمى" **Mode Of Operation** أنماط أو أساليب العمليات" والبعض يطلق عليها **Feedback Modes** سنتناول هذه الأساليب في النسخة النهائية من الكتاب .

على العموم اشهر هذه الأساليب هو **cipher block chaining** اختصارا (**CBC**) ، هنا في هذا النمط سوف نطبق العملية XOR في النص الأصلي الحالي و النص المشفر السابق .

كتله النص الأصلي الحالي **XOR** كتله النص المشفر السابقة ، وبعدها سوف نجري عمليه التشفير .

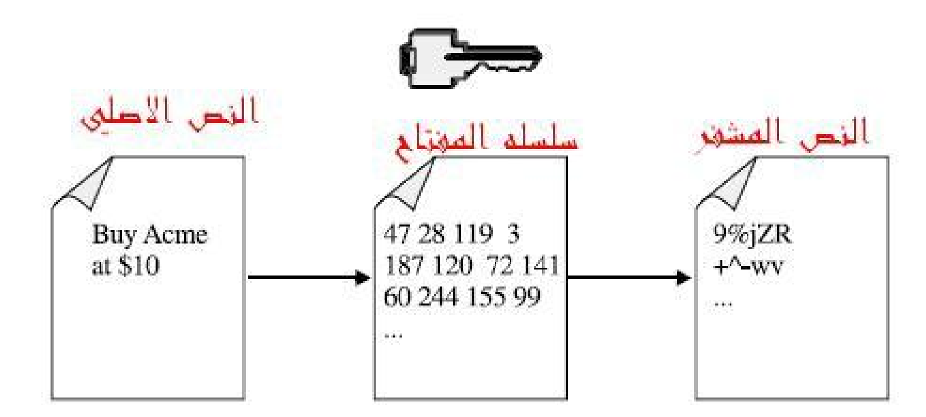
بالنسبة إلى كتله النص الأول ، لن يكون هناك نص مشفر سابق لذلك سوف نطبق العملية مع متغير اسمه ***initialization vector*** اختصارا **IV** .



وهكذا سوف ننهي مشكله تكرار البيانات بهذه الطريقه .

شفره التدفق *Stream Ciphers*

النوع الثاني من أنواع التشفير بالمفتاح المتناظر هو شفره التدفق ، وهنا سوف نتعامل مع بت بت أو بايت بايت وليس كتله كتله ، وعمليه توليد المفتاح **Key Stream** من الممكن أن تعتمد على النص المشفر السابق ومن الممكن لا (هناك نوعين من هذا النوع ، نوع متزامن ونوع غير متزامن) .



Block VS Stream من الأفضل

شفره التدفق هي أسرع بكثير من شفرات الكتل وعمليه كتابه برامج سهله وأكودها اقل بكثير من الكتل ، واحد اشهر أنواع شفرات التدفق RC4 وهي أسرع بكثير من أي نوع من أنواع شفرات الكتل ، وتتطلب حوالي 30 سطر فقط في الكود . معظم شفرات الكتل تأخذ على الأقل 200-400 سطر.
شفرات الكتل من جهة أخرى تسمح باعاده استخدام المفتاح ، بعكس شفرات التدفق التي تستخدم المفتاح مره واحده فقط ، في الكثير من الأحيان يجب أن نشفر العديد من الأشياء بمفتاح واحد.

مثال ، شركه لديها قاعدة بيانات ضخمه للعملاء تحتوي معلوماتهم من أرقام هواتف وبطاقات ائتمانية وغيرها ، في حال استخدمت شفرات التدفق سوف تتطلب لكل مدخل (عميل) مفتاح خاص وهذا يتطلب مئات من المفاتيح وهو أمر غير عملي ، أما في حاله استخدمت شفرات الكتل فإنها تشفر جميع البيانات باستخدام مفتاح واحد ، ولفك تشفير بيانات أي عميل نستخدم نفس المفتاح . عمليه اداره المفتاح أسهل بكثير في هذه الحالة.

لذلك في معظم قواعد البيانات يتم استخدام شفرات الكتل Block Cipher وأيضا في برامج البريد الالكتروني ، وأيضا في برامج تشفير الملفات.

Digital Encryption Standard

في بدايه السبعينات تم معرفه انه اغلب الشفرات القديمة لم تعد مجديه وغير نافعة للتشفير ، ولهذا قرر علماء في شركه IBM بعمل خوارزمية جديدة للتشفير تبنى على بنية قديمة تسمى Lucifer (نسبه إلى مخترعها Horst Feistel) ، ومن خلال مساعده وكاله الأمن القومي NSA تم عمل خوارزمية DES .

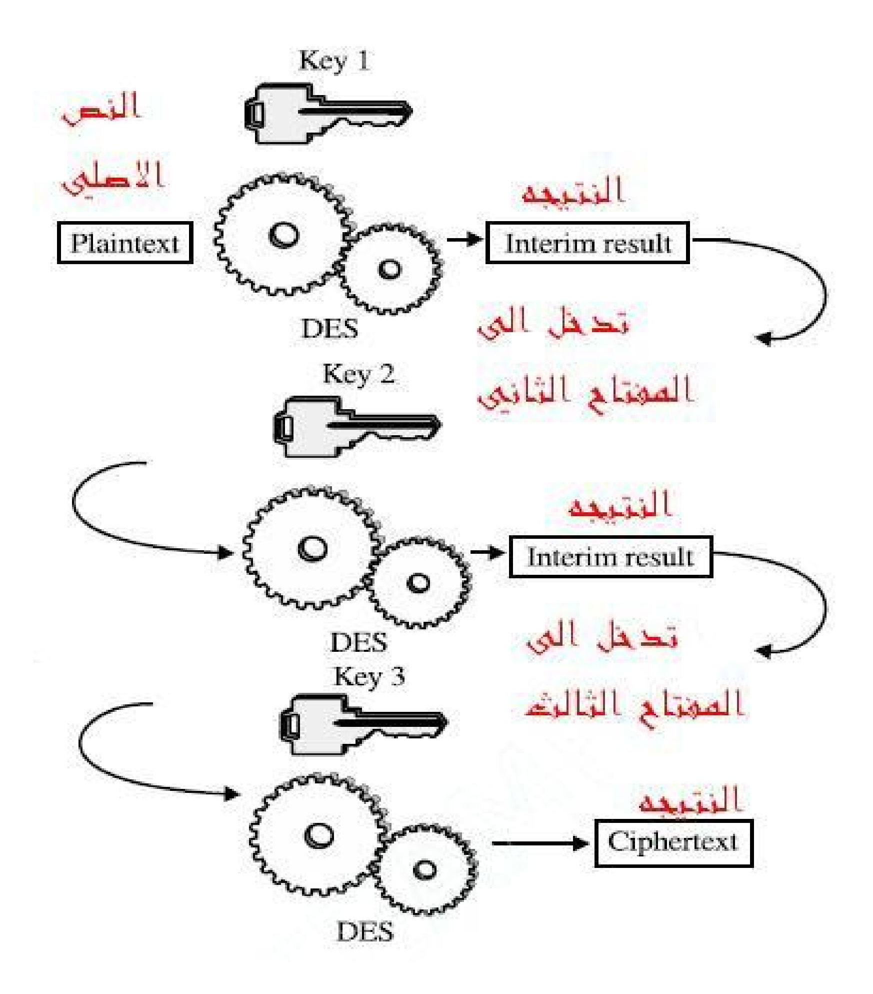
DES هي احد شفرات الكتل Block Cipher ، وتأخذ مفتاح بطول 56 بت ، وتعمل على كتله طولها 64 بت .

وفي الثمانينات لم يتم اكتشاف أي ثغره في DES لذلك كانت اقوي الخوارزميات في ذلك الوقت، ولكسر أي رسالة مشفره بها لم يكن هناك إلا استخدام هجوم الـ brute-force ، ولأن طول المفتاح 56 بت (مداه من 0 إلى 72 كوارلديون) و الاجهزه بطيئة للغاية ، فكانت عمليه الكسر تتطلب سنه كاملة.

وفي 1999 وفي احد المؤتمرات تم كسر هذه الخوارزمية في 24 ساعة من قبل the Foundation Electronic Frontier اذا العالم يجب أن ينتقل إلى خوارزمية أخرى .

Triple DES

احد البدائل كانت خوارزمية Triple DES أو البعض يسموها 3DES ، هي بكل بساطه DES ولكن ثلاثة مرات ، يعني سوف تدخل الكتلة الأولى (16 بايت) إلى الخوارزمية بالمفتاح الأول ، والناتج سوف يدخل إلى الخوارزمية مع المفتاح الثاني ، والناتج سوف يدخل مع المفتاح الثالث .



هنا سوف نستخدم ثلاثة مفاتيح ، كل منها بطول 56 بت (أي كأنها 168 بت) ، قد تسأل وتقول **"اذا كانت كسر مفتاح واحد يأخذ 24 ساعة ، اذا كسر ثلاثة مفاتيح سوف يأخذ 72 ساعة"** هل هذا صحيح ؟

بالطبع لا ، واليك مثال بسيط :

لنفترض أن المفاتيح الثلاثة A , b , c كل منها مداه من 0 إلى 72.. quadrillion
وأن المفتاح هو :

A = 1
B = 33,717
C = 1,419,222

الآن المخترق سوف يجرب a=0 , b=0 , c=0 وهو ليس المفتاح الصحيح..

بعدها سوف يجرب المفتاح a=1, b=0 , c=0 وهو ليس المفتاح الصحيح ..
(مع العلم بأن الأول a صحيح) لكن كيف يعرف المخترق ذلك ؟ النتيجة لم تظهر صحيحة إلا اذا كانت الثلاثة مفاتيح صحيحة والا فسوف يدور في حلقه مفرغه.

على العموم 3 DES لها مشكله وهي أنها بطئيه جدا ، الـ DES العادية هي بطيئة ، فما بالك بـ DES ثلاث مرات ، واغلب التطبيقات تتطلب السرعة في العمل ، وهذه الخوارزمية لا تنفع لأنها بطيئة جدا ، اذا العالم مره أخرى بحاجه إلى خوارزمية!!

البدائل:
بعد مشكله البطء في خوارزمية الـ 3 DES ، اتجه العديد من الأفراد والشركات لتطوير خوارزمياتهم الخاصة وكانت النتيجة أن هناك العديد من الخوارزميات الجيدة والتي تأخذ مفاتيح متغير الطول (وليس ثابتة الطول كما في DES) ، من هذه الخوارزميات
. RC2, RC5, IDEA, CAST, SAFER, Blowfish
هذه الخوارزميات بالرغم من قوتها لم تصبح واسعة الانتشار كما في DES و Triple DES .
اذا العالم بحاجه هذه المرة إلى مقياس أو خوارزمية واسعة الانتشار كـ DES

Advanced Encryption Standard

نتيجة لهذا الأمر قام المعهد الوطني للمعايير National Institute of Standards and Technology اختصارا NIST ، باستدعاء جميع المهتمين بهذا الأمر وكلفت بكل منهم بعمل خوارزميته الخاصة وفي النهاية أقوى خوارزمية سوف تكون هي المقياس الجديد AES ، وقد قدمت 15 خوارزمية (منها القوي ومنها الضعيف) .

وفي 1999 قامت NIST باختيار أفضل 5 خوارزميات بعد إجراء العديد من الاختبارات ، وقد جعلت الأمر بالتصويت لأفضل خوارزمية ، وفي 2000 تم اختيار خوارزمية **Rijndael** كالمقياس الجديد AES .

اداره المفتاح المتناظر *Symmetric-Key Management* :

توصلنا سابقا إلى أن التشفير بالمفتاح المتناظر يقوم بتشفير الرسالة بمفتاح ما ، ثم يقوم بفك التشفير بنفس المفتاح ، لذلك عمليه الحفاظ على المفتاح أمر في غاية الاهميه ، فإذا انكشف المفتاح انكشفت جميع الأسرار ، لذلك يجب حفظ المفتاح في مكان امن جدا ، عمليه الحفاظ على المفتاح تسمى بـ **اداره المفتاح Symmetric-Key Management** .

ربما الآن تتساءل"اذا كان هناك مكان أستطيع أن احفظ في المفتاح ، فلماذا لا احفظ الرسالة في ذلك المكان ولا احتاج إلى التشفير" ؟

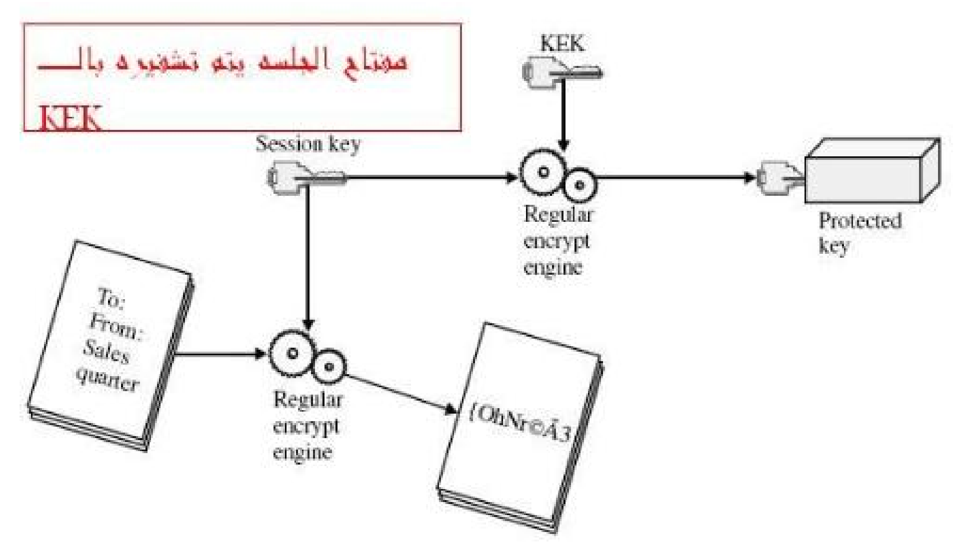
في الحقيقة حفظ مفتاح التشفير (56 بت مثلا) يكون أسهل كثيرا من حفظ الرسالة (بعض الأحيان حجمها يكون مئات من الميغا بايت MB) ، بالاضافه إلى هناك حلول لحفظ المفتاح عن طريق حفظها داخل أجهزه صغيره مصممة لهذا الغرض.

Password-Based Encryption

إن المفتاح الذي كنا نستخدمه للتشفير وفك التشفير يسمى في الحقيقة "**بمفتاح الجلسة session key**" ، وأحد الطرق لحماية هذا المفتاح هي عن طريق تشفيره أيضا ، أي أن المفتاح (مفتاح الجلسة) يحتاج إلى مفتاح آخر لكي يتم تشفيره.

هذه العملية تسمى بالـ **password-based encryption** واختصارا **PBE** .

يعني مفتاح الجلسة session key هو المفتاح الذي نستخدمه في التشفير وفك التشفير
ومفتاح تشفير المفتاح key encryption key هو المفتاح الذي نستخدمه لتشفير مفتاح الجلسة ، واختصارا يسمى KEK .



الآن بما أن المفتاح KEK (من الآن وصاعدا نسميه بهذا الاسم) هو الذي يستخدم لتشفير وفك تشفير مفتاح الجلسة ، السؤال هل أنا بحاجه إلى حماية هذا الـ KEK ؟
الجواب ، هو لا ، عندما نريد أن نشفر المعلومات نقوم بتوليد هذا المفتاح (بأحد طرق توليد الأرقام العشوائية) بعدها نقوم باستخدامه ومن ثم نحذفه ، وفي حاله فك التشفير نقوم بتوليد هذا المفتاح مره أخرى ونستخدمه ومن ثم نحذفه ، وفي مرحله توليد هذا المفتاح يجب أن ندخل باسورد معين سواء في مرحله التشفير أو فك التشفير .

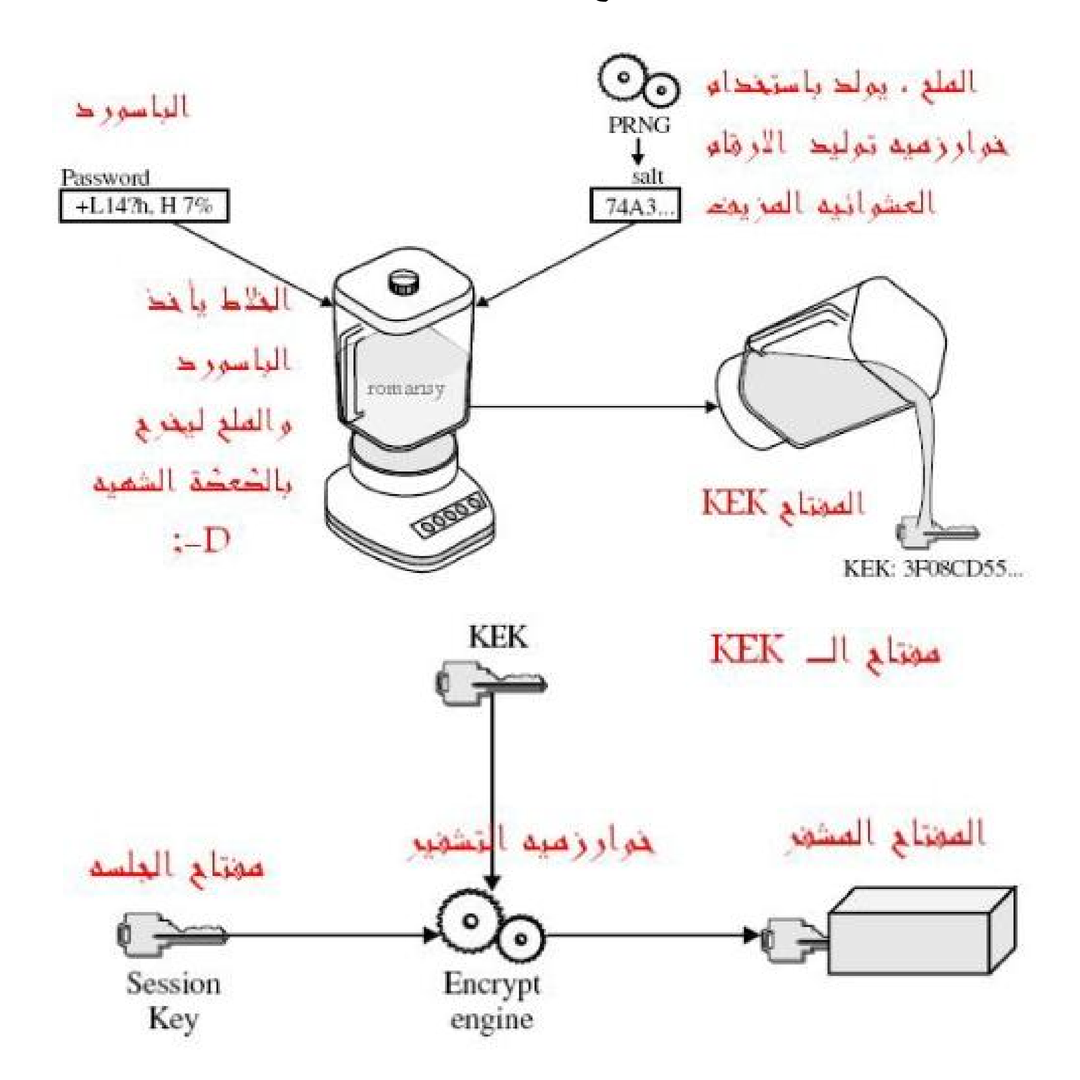
بصوره مبسطه ، **مفتاح الجلسة Session key** هو الذي يشفر المعلومات ونقوم بتوليده عشوائيا .
مفتاح الـ KEK هو الذي يشفر مفتاح الجلسة ونقوم بتوليده عن طريق password-based encryption.

ويتم توليد الـ KEK :
-1- إدخال باسورد
-2- استخدام أي طريقه لتوليد أرقام عشوائية لتوليد الـ salt (الملح) .
-3- ندخل الباسورد والملح مع بعض داخل الخلاط blender والناتج هو خليط من البتات العشوائية ، لقد تطرقنا سابقا عن الـblender وسوف نتحدث عنه بالتفصيل لاحقا .
-4- نأخذ ما يكفي من الخليط السابق ونضعه داخل المفتاح KEK ، وبعدها نستخدم الـ KEK لتشفير مفتاح الجلسة ثم نحذف هذا الـ KEK ، ونحتفظ بالملح .
-5- الآن تم تشفير الرسالة ، ويجب آن تحفظ الملح لأنه سوف يستخدم في فك التشفير .

ما هو هذا الملح ، بالتأكيد هو ليس الذي تستخدمه في الطعام ، وسوف نتطرق له بعد قليل .

الآن لفك التشفير:

-1- ندخل الباسورد الذي أدخلناه في عمليه التشفير
-2- نأتي بالملح الذي احتفظنا به في مرحله التشفير
-3- ندخل الملح والباسورد في نفس الخلاط الذي استخدمناه في عمليه التشفير ، في حال اختلف احدهم سوف يكون الناتج عبارة عن KEK خاطئ ، وفي حال كانوا صحيحين فالناتج هو الـ KEKالصحيح
-4- نستخدم الـ KEK لفك مفتاح الجلسة ، وبعدها نستخدم مفتاح الجلسة لفك تشفير الرسالة..
وانتهت الخطوات ، والصورة التالية توضح العملية :



دعنا نوضح بعضا من النقاط و الاسئله التي ربما ستتساءل عنها :

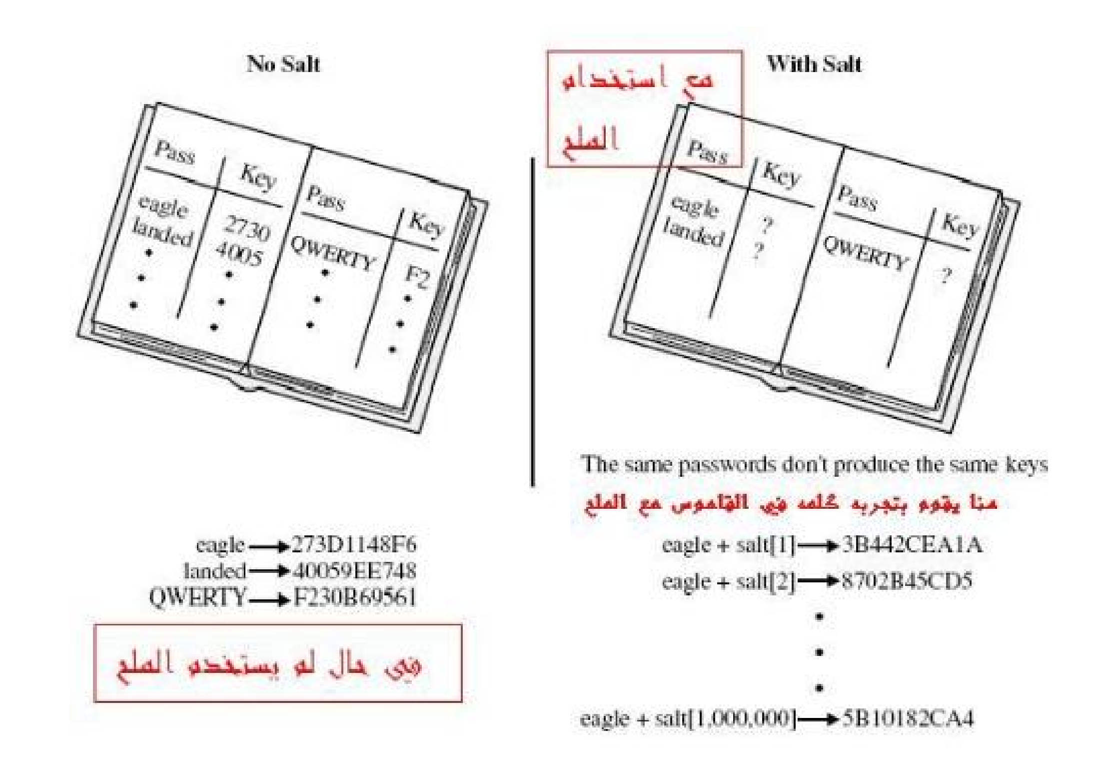
Mixing Algorithms and KEK لماذا نخلط بين الباسورد والملح ؟ لماذا لا يكون الباسورد هو الـ KEK

الجواب ، لان الباسورد لا يحتوي على الكثير من Entropy ، لذلك هو غير كافي ابدآ ، فهنا نستخدم هذه الخوارزمية للخلط بين الملح والباسورد وبذلك النتيجة أكثر عشوائية..

ما أهميه هذا الملح The Necessity of Salt ؟

هذا الملح ببساطه يستخدم لمنع محاولات تخمين الباسورد ، لأنه في حاله استخدمنا الباسورد فقط ك KEK فان المخترق بمكانه أن ينشئ قاموس به كل اغلب الباسوردات والمفاتيح وبعدها يبدأ في البحث عن الباسورد الخاص بك لكي يعرف الـ KEK (هذا الهجوم يعرف بـ dictionary attack) .

لكن اذا استخدمنا الملح ، فان المخترق لكي يعرف الـ KEK يجب في البداية أن يعرف هذا الملح للباسورد الفلاني هل هو صحيح أم لا ، اذا كان غير صحيح يقوم بتجربة ملح أخر في نفس الباسورد السابق ، اذا كان غير صحيح يقوم بتجربة ملح آخر في نفس الباسورد ، بعدها يغير الباسورد ويجرب الأملاح (☺) مره أخرى ، وهكذا يكون الأمر طويل جدا جدا جدا..



الآن وبعد تشفير مفتاح الجلسة باستخدام مفتاح KEK ، هل تعتبر في أمان كامل من جميع الهجمات ؟

بالطبع لا ، لان المخترق بإمكانه عمل هجوم على المفتاح KEK (هجوم القوه العنيفة Brute Force attack) ويقوم بتجربة مفتاح مفتاح إلى أن يصل إلى المطلوب .

أو بإمكانه عمل هجوم على الباسورد (Brute Force Attack) ، ويقوم بإدخال الباسورد والملح في الخلاط ، بعدها يأخذ الناتج KEK ويفك تشفير مفتاح الجلسة وبعدها يفك تشفير البيانات ، وإذا لم يصلح الباسورد يقوم بتغييره واختيار واحد أخر .

قد تبدو العملية طويلة ، لكن في الحقيقة أساليب هجوم Brute Force قد تأخذ أساليب متطورة ، مثلا عمل البرنامج **بالتوازي in parallel** وهنا سوف يستفيد من عمل المعالج بشكل كبير ، أيضا من الممكن أن يعمل أكثر من جهاز في عمليه الكسر .

أيضا من الممكن أن يقوم المخترق باستخدام **هجوم القاموس dictionary Attack** وهنا يقوم بعمل قاعدة بيانات لأغلب الباسوردات في جميع اللغات ، بعدها يقوم بتجربة هذه الباسوردات .
وهذا الهجوم بالطبع أسرع من هجوم الـ Brute Force لأنه يكون محدود على مجموعه من الباسوردات .

لتفادي أنواع الهجوم في مرحله اختيار الباسورد يجب اختيار باسورد قوي **مثل**
14G:c*%3└ wM*-l6g]_Bnp?└ d86

الباسورد السابق جيد جدا بل ممتاز ويصعب تخمينه ، لكن من الذي يستطيع أن يحفظ مثل هذا الباسورد ؟ لا أحد بالطبع ، على العموم اذا كانت لديك القدرة في حفظ مثل هذا الباسورد ، فلن تكون لديك أي مشكله بعد الآن .

و التوصيات والمقالات بشأن اختيار كلمات السر هي كثيرة ، واشهر الخطوات هي أن يكون مكون من عده أرقام وحروف بشكل مختلط وعلى اقل تقدير 10 حروف ، أيضا استخدام باسورد مختلف لكل حساب ، وعدم استخدام كلمات معروفه مثل الأسماء وأرقام الهاتف أو ما شابه .

أجهزه حفظ المفاتيح *Hardware-Based Key Storage*

لقد رأينا قبل قليل أن احد طرق حفظ المفتاح هو استخدام الـ PBE ، احد الحلول الأخرى هي استخدام أجهزه خاصة لحفظ المفاتيح ، بعض هذه الاجهزه صغير جدا ويسمى **Token** ، وبعضها كبير ويسمى **crypto accelerators** .

ال Tokenبعض الأحيان تكون بطاقة بلاستيكية ، أو بطاقة ذكيه (تحتوي على جهاز داخلها) ، أو حتى خاتم صغير ، أو جهاز يوصل بفتحه USB ، هذه الـ Token تحتوي على معالج صغير بداخلها ، وذاكره صغير الحجم طبعا المواصفات بينها تختلف على حسب الشركة المصنعة لها ، بعض هذه ال Token يأتي بذاكره كبيرة مثل نوع اسمه : 1970s era PC .





Smarty smart card and reader



RSA SecurID 3100 smart card

الميزة الاساسيه في هذه الـ Token هو أن المخترق لا يستطيع الوصول إليها ، وحتى إن استطاع سرقه احدها فإنها تطالب بباسورد ورقم PIN ، وفي حال اخطأ المخترق إدخال الرقم عده مرات سوف يبدأ نظام الحذف تلقائي ، يعني يصعب جدا اختراقها .

احد الاستخدامات الأخرى لهذه الـ Token بالاضافه إلى حفظ المفاتيح ، هي حفظ الباسوردات ، مثلا كان لديك عده باسوردات لعده مواقع ، للايميل والشات والمنتدى وكل باسورد مختلف عن الآخر ، أسهل طريقه هي حفظ هذه الباسوردات داخل الـ Token ، وعندما تريد الدخول مثلا إلى بريدك قم بتركيب جهاز الـ Token بعدها تأخذ الباسورد المطلوب.

Crypto Accelerators

الاجهزه الكبيرة التي تستخدم في عمليه حفظ المفاتيح تسمى بـ **crypto accelerators** ، هذه الاجهزه لها ميزه جيده جدا ، وهي تحتوي على معالج ذكي خاص بها ، في حال تم سرقه احد هذه الاجهزه وقام المخترق بمحاوله فتح الجهاز أو استخدام أي طريقه من طرق استعاده البيانات ، فيقوم المعالج بحذف جميع البيانات والمفاتيح.



أيضا هذه الاجهزه تأتي بمولدات للأعداد العشوائية الصحيحة RNG وبعضها يأتي مولدات لأعداد عشوائية مزيفه PRNG ، لكن تكون تهيئه الـ Seed من المصنع (لكنها في النهاية Pseudo) .

Biometrics

الاجهزه السابقة التي ذكرناها سواء Token أو crypto accelerators تعتبر خيار ممتاز في عمليه حفظ المفاتيح و الباسوردات ، لكن في الحقيقة هذه الاجهزه لا تكفي لضمان الأمن التام ، وبالذات في الجهات التي يراد الحفاظ على المعلومات بصوره تامة ، لان هذه الاجهزه قد تسرق ويسرق الباسورد وبعدها يصل المخترق إلى البيانات .

لذلك تم اختراع الـ **Biometrics** وهي عده طرق تستخدم للتأكد من الشخص الذي يحاول الوصول إلى البيانات هو الشخص المسموح له ، مثلا جهاز التأكد عن طريق بصمه الإصبع ، وطبعا من المستحيل أن تكون هناك بصمتين متشابهتين (حتى في التوائم) ، بعض هذه الاجهزه تستطيع معرفه الإصبع هل هو من إنسان حي أم من إنسان ميت أو إصبع مقطوع!

أيضا هناك أجهزه لمسح شبكيه العين ، وأجهزه للتعرف على نبره الصوت ، وأجهزه للتعرف عن طريق الـ DNA وغيرها ، على العموم هذه الاجهزه غير منتشرة بشكل كبير (حاليا) بسبب

تكلفتها العالية ، وأيضا بسبب أنها غير موثوقه بشكل أكيد ، يعني مثلا اذا أصيب الإصبع بجرح هل يستطيع الجهاز التعرف عليه ، في حال استخدام نبره الصوت وأصيب الشخص بنوبة برد أو التهاب في الحلق ، هل يستطيع الجهاز التعرف على الصوت!

بالتأكيد مع تطور العلوم وخاصة الذكاء الاصطناعي فانه قد يأتي يوما يتم زرع جهاز كمبيوتر شخصي داخل دماغ أي إنسان ويكون التعرف عن طريق قرائه هذا الجهاز (قد قرأت من فتره عن أمر مشابه من شركه مايكروسوفت) .

The Key Distribution Problem and Public-Key Cryptography

تعرفنا قبل قليل على بعضا من أساليب حماية المفتاح (مفتاح الجلسة) ، ويكون إما عن طريق تشفيره مره أخرى (PBE) ، أو عن طريق تخزين المفتاح في احد الأجهزة الخاصة لذلك Token، إلى هنا الأمر تحت السيطرة ، لكن ماذا اذا أردنا أن نرسل المفتاح إلى شخص آخر حتى يقوم بفك تشفير الرسالة التي سوف أرسلها له (تذكر أن التشفير بالمفتاح المتناظر ، المفتاح نفسه يقوم بالتشفير وفك التشفير).

بعبارة مبسطه ، في حال قمت بتشفير رسالة ما بهذا المفتاح المتناظر ، بعدها أرسلت الرسالة إلى الشخص الذي أريد ، في حال وصلت الرسالة للشخص هذا سوف تكون غير مفهومه وذلك لان المفتاح الذي يفك التشفير معي والى الآن لم أرسله للشخص المراد ، أيضا في حال كان هناك مخترق ووصلت الرسالة إليه بطريقه ما (سواء قام باختراق جهاز الشخص الذي أرسلت له الرسالة ، أو قام بالتقاط الرسالة أثناء إرسالها) المهم سوف تكون الرسالة أيضا غير مفهومه لأنه لا يملك المفتاح.

اذا السؤال هنا ، كيف يمكن أن أرسل المفتاح بطريقه آمنه إلى الشخص الذي أريد ، وفي نفس الوقت لا يستطيع المخترق الحصول عليه ؟؟
هذه المشكلة تسمى بمشكله **إرسال المفتاح Key Distribution Problem** ، والتي بسببها تم اختراع الطريقه الأخرى في التشفير وهي التشفير بالمفتاح غير المتناظر Asymmetric key Cryptography .

قبل الخوض في التشفير بالمفتاح غير المتناظر وجدت بعضا من الحلول التي قد تكون مناسبة لحالتك (أي قد تستطيع الاكتفاء بها) ، هذه الحلول هي :
-إرسال المفتاح قبل عمليه الإرسال
-استخدام طرف ثالث موثوق Trusted Third Party ، اختصاراTTP

نبدأ بالحل الأول : إرسال المفتاح قبل عمليه الإرسال :
وهنا في هذه الحالة يجب أن أقوم بإرسال المفتاح إلى الشخص المراد قبل أن ابدأ في إرسال الرسالة ، ولكن بالطبع يجب أن أتأكد أن الاتصال امن حتى أرسل المفتاح وأنا مطمئن انه لا يوجد احد غير الشخص المراد إرسال المفتاح إليه .
حسنا أأمن طريقه لإرسال المفتاح ، هي القيام بإعطاء المفتاح للشخص وجها بوجه ، أي أن اذهب إلى مكتبه أو بيته ، بعدها أقوم بتوليد المفتاح واحفظ نسخه في جهازي ، ونسخه في جهاز

الشخص المراد . هكذا أكون متأكد أن المخترق لا يستطيع معرفه المفتاح لأنه حاليا لا يوجد (إلا اذا كان واضع بعضا من برامج التروجان key logger وبهذا قد يعرف المفتاح ، لكن نستبعد هذا الأمر بفرض انه أجهزتنا محمية من التروجنات) .

الآن عندما أريد أن أرسل رسالة ما لصديقي ، أقوم بتشفير الرسالة بالمفتاح المتناظر ، وأرسلها له وسوف يقوم بفك تشفيرها بالمفتاح المتناظر الذي أعطيته إياه أثناء اللقاء.

حسنا ، ماذا لو كان الشخص في بلد بعيد أو على اقل تقدير لا أستطيع اللقاء به ؟
بالطبع لن أرسل المفتاح عن طريق الانترنت لأنها شبكه غير محمية واحتمال تواجد المخترق لالتقاط المفتاح كبير جدا ، اذا ما الحل في هذه الحالة ؟

هنا أقوم بتشفير المفتاح المتناظر عن طريق الـ KEK (طبعا لازم ادخل باسورد معين) ، بعدها أقوم بإرسال المفتاح الناتج إلى صديقي عن طريق الانترنت ، في حال حصل المخترق على المفتاح المشفر فإنه لا يستطيع الحصول على مفتاح الجلسة لأنه بكل بساطه يوجد باسورد وهو لا يملكه ، أيضا صديقي نفس الحالة يجب أن يحصل على الباسورد حتى يفك تشفير المفتاح وبالتالي يفك تشفير الرسالة ، كيف يمكن أن أرسل الباسورد ؟؟

هنا وبكل بساطه يمكنك إرساله عن طريق مثلا الهاتف ، وهكذا لن يحصل عليه المخترق . اذا كنت تعتقد انك مراقب بكل الوسائل الممكنة (الهاتف ، الانترنت ، المقابلة الشخصية) فبالتأكيد هذه الطريقه لن تصلح لك ويجب أن تحل المشكلة بطريقه أخرى ، ولكن في حال كان الهاتف غير مراقب ، فهي تعتبر كافيه ومناسبة !

في حال كان إرسال المفتاح قبل الإرسال مناسب لك ، ماذا اذا كنت تريد إرسال المفتاح إلى عده أشخاص بدلا من شخص واحد ؟

العملية سوف تتشعب كثيرا لأننا في هذه الحالة سنحتاج إلى الذهاب إلى مكتب كل عميل ونقوم بتوليد المفتاح ، كل شخص يتوقع رسالة من شخص ما في الشركة يحتاج إلى أن يذهب إلى ويقوموا بتوليد مفتاح ، وهو حل غير عملي بتاتا ، لكن يمكن أن تقوم الشركة بعمل اجتماع الغرض منه تبادل المفاتيح بين الأشخاص ، وهو حل جيد ، لكن تبقى هناك مشكله قدوم عميل جديد ، ما الذي سوف يحصل هل يقوم جميع العملاء بزيارة العميل الجديد ويقوم بتبادل المفاتيح ، وهي مشكله أيضا .

اضافه في حال زيادة عدد الأشخاص في عمليه تبادل المفاتيح سوف يكبر عدد الزيارات بشكل كبير ، في حال نحن اثنين سوف يكون هناك لقاء واحد (أو اتصال واحد) ، في حال كنا ثلاثة سوف نحتاج إلى لقاءين (اذهب أنا إلى الشخص الأول ونولد مفتاح ، بعدها اذهب إلى الشخص الثاني وأعطيه المفتاح) ، في حال كنا أربعه فسوف نحتاج إلى ستة لقائات ، بشكل عام:
مجموعه من الأشخاص مكون من n عدد ، اذا يحتاجون إلى **$1/2 * (n^2 - n)$** لقاء..

مثلا 10 أشخاص ، 100-10 = 90 ، 90/2 = 45 ، اذا نحتاج إلى 45 لقاء ، اذا كان هناك 20 شخص سوف نحتاج إلى 190 عمليه تبديل مفاتيح ، في حال كانت الشركة بها 1000 عميل ، سوف نحتاج إلى 499500 عمليه تبديل !!

يمكن حل هذه المشكلة ، عن طريق جعل جميع الأشخاص يستخدموا مفتاح واحد ، لكن في حال مثلا خروج أحد العملاء منها فسوف تكون هناك مشكله لأنه يعرف المفتاح ، لذلك على الشركة أن تقوم بتغير المفتاح ، وإعطاء جميع العملاء المفتاح الجديد (من خلال مثلا الاجتماع) وهو حل "يبدو" جيد .

وتبدأ المشاكل في حال تمكن المخترق من كسر رسالة والحصول على المفتاح ، سوف يكون بإمكانه كسر جميع الرسائل (في حال كان طول المفتاح 128 بت والخوارزمية جيده ، فانه لن يمكن كسر المفتاح بسهوله) لكن اذا افترضنا انه قام بسرقة المفتاح (مثلا أخذه من عميل غشاش ، أو استخدم أسلوب التهديد) المهم حصل عليه ، فهنا يكون المخترق قد حصل على كشف الرسائل لأنه يملك المفتاح . وهذا ما لا يريده أحد ، لذلك حل "مفتاح واحد للجميع" حل غير عملي أيضا.

باختصار يكون حل إرسال المفتاح قبل عمليه الإرسال مناسب اذا كانت عمليه الإرسال بسيطة (بين شخصين أو ثلاثة مثلا) ، وإذا كانت أيضا عمليه الإرسال عبر الهاتف آمنه .

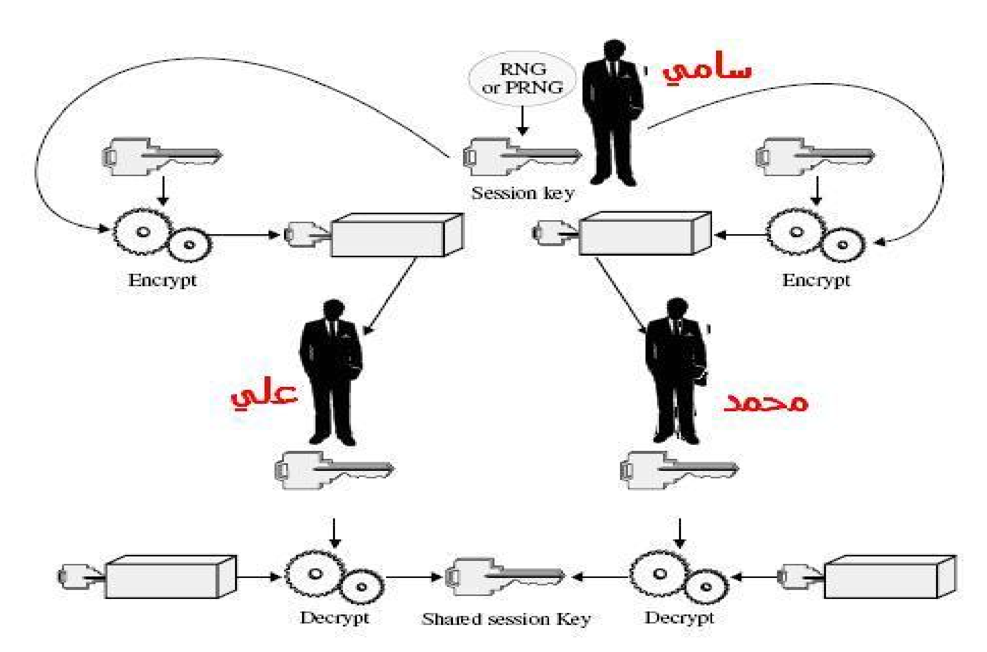
غير ذلك سوف نلجأ إلى الحل الثاني، و هو استخدام طرف ثالث في العملية TTP :

دعنا نوضح الآمر بمثال بسيط .

لنفرض محمد يريد أن يرسل رسالة لعلي ، والطرف الثالث هو سامي .

الآن في البداية يذهب محمد إلى الطرف الثالث سامي ويقوم سامي بتوليد مفتاح KEK ويقوم بتخزين المفتاح في جهازه ، واعطاء نسخه من الـ KEK إلى محمد. يأتي علي أيضا إلى الطرف الثالث سامي ، ويقوم بتوليد مفتاح KEK ويخزنه في جهازه ، ويعطي على نسخه من المفتاح.

هنا عندما يريد محمد إرسال رسالة إلى علي ، يقوم محمد بطلب مفتاح الجلسة من الطرف الثالث سامي ، يقوم سامي بتشفير مفتاح الجلسة بالمفتاح KEK الذي قام بتوليده هو و محمد في البداية ، ويقوم بإرسال المفتاح المشفر إلى محمد ، ويقوم بتشفير مفتاح الجلسة أيضا مره أخرى ولكن بمفتاح علي ويقوم بإرساله إلى علي.



هنا في هذه العملية سامي هو الوسيط بين الاثنين ، هو الوحيد الذي يملك مفتاح الجلسة ، وهو الوحيد الذي يمكنه قراءه الرسائل ، لذلك يجب أن يكون طرف موثوق ، والا سوف تنكشف كل الرسائل ، ولأن الطرف الموثوق قد يكون غير موجود بالنسبة لأغلبنا فهذا الحل غير مجدي للكثير منا ، ويجب أن نحل المشكلة بطريقه أفضل..

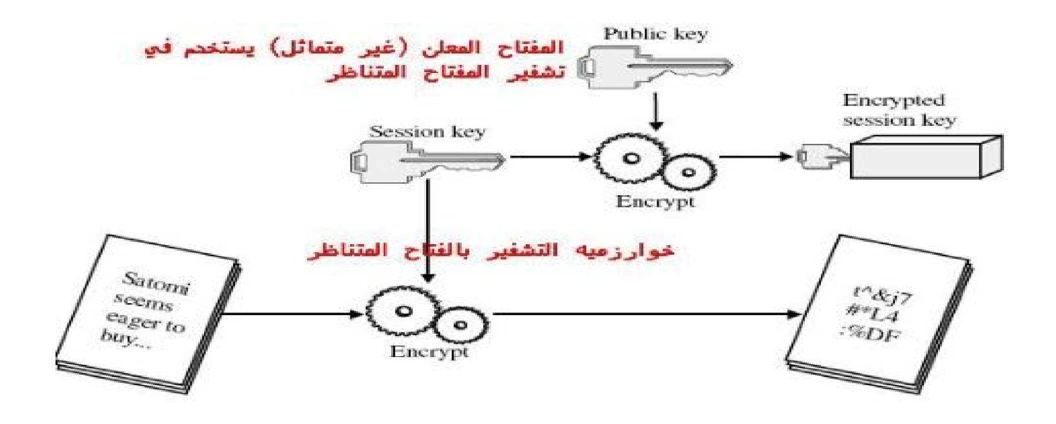
ومن هنا جاءت الطريقه الأخرى للتشفير وهي التشفير بالمفتاح الغير متناظر **Asymmetric key Cryptography**، بالمناسبة اغلب الكتب تسمى هذه الطريقه باسم آخر وهو التشفير بالمفتاح العام (أو المعلن **Public-Key Cryptography**)، على العموم الاسمين يؤديان نفس الغرض .

Public-Key Cryptography and the Digital Envelope التشفير بالمفتاح العام ، والظرف الرقمي :

في السبعينيات تم اختراع تلك الطريقه ، وهي تستخدم مفتاحين ، **مفتاح عام public Key** للتشفير ، **ومفتاح خاص private Key** لفك التشفير ، مثلا اذا أراد محمد إرسال رسالة مشفره لعلي باستخدام التشفير بالمفتاح الغير متناظر (أو المعلن أو العام) ، يقوم محمد بأخذ المفتاح العام من علي ، وبعدها يقوم بتشفير الرسالة بهذا المفتاح ، بعدها يقوم بإرسال الرسالة إلى علي الذي يقوم بفك التشفير بالمفتاح الخاص به..

المفتاح العام public Key يكون معروف للجميع وأي شخص يستطيع الحصول عليه (هو يستخدم فقط للتشفير) .
المفتاح الخاص private Key يكون غير معروف (معروف لشخص واحد) وهو يستخدم لفك التشفير
ولكي نكون أكثر صوابا ، فان المفتاح العام يستخدم لتشفير مفتاح الجلسة ؟؟ ما هذا الكلام !!
دعنا نعيد المثال مره أخرى...

محمد يريد إرسال رسالة لعلي ، ويريد طريقه آمنه لإرسال المفتاح ، لذلك تم اختيار طريقه التشفير بالمفتاح الغير متناظر ، بعدها يقوم محمد بتشفير الرسالة بخوارزمية التشفير بالمفتاح المتناظر (لاحظ المتناظر ، سأذكر السبب بعد قليل) ، وبعدها يقوم بأخذ المفتاح العام لعلي (الذي هو معروف للجميع) ويقوم بتشفير المفتاح (الجلسة) بعدها يرسل هذا المفتاح المشفر والرسالة إلى علي ، الذي يقوم بدوره بفك تشفير المفتاح بالمفتاح الخاص ، بعدها يقوم بفك تشفير الرسالة..



لماذا قمنا بتشفير الرسالة بخوارزمية التشفير بالمفتاح المتناظر ؟

ولذلك بسبب **السرعة والأداء** ، فالتشفير بالمفتاح المتناظر أسرع بكثير من التشفير بالمفتاح الغير متناظر ،

التشفير بالمفتاح المتناظر يشفر 50 MB في الثانية الواحدة

التشفير بالمفتاح غير المتناظر يشفر 20-200 KB في الثانية الواحدة . –لاحظ الفرق- .

لذلك يستخدم المفتاح المتناظر لتشفير الرسالة ، ويستخدم المفتاح الغير متناظر(العام) لتشفير المفتاح المتناظر ، العملية السابقة تسمى بالظرف الرقمي Digital Envelope .

هذه العملية مشابه لعمليه Password Based Encryption ، لأنها في PBE نقوم باختيار خوارزمية لتشفير النص من نوع Symmetric ، بعدها نقوم بتوليد مفتاح الجلسة ، ونقوم بتشفير مفتاح الجلسة باستخدام PBE ، ولا يمكن لأي أحد فك تشفير مفتاح الجلسة إلا في حال كان يملك الباسورد الصحيح .

أما هنا في Digital Envelope ، نقوم باختيار خوارزمية لتشفير الرسالة من نوع Symmetric ، ونقوم بتوليد مفتاح الجلسة ، بعدها نحصل على المفتاح العام من الطرف الأخر الذي أريد إرسال الرسالة إليه ، وأقوم بتشفير مفتاح الجلسة بهذا المفتاح العام ، وأقوم بإرسال النص المشفر (بـSymmetric) أضافه إلى مفتاح الجلسة المشفر (بالمفتاح العام) للطرف الأخر ، وهنا يقوم الطرف الأخر باستلام الرسالة ، ويقوم بفك تشفير مفتاح الجلسة باستخدام المفتاح الخاص به ، وبعدها يحصل على مفتاح الجلسة ، ومنه يقوم بفك تشفير الرسالة .

لاحظ أن هذه الطريقه قد حلت مشكله توزيع المفاتيح ، حيث كل ما الأمر استخدام طريقه public Key Cryptography ، وبعدها لا حاجه لطرف ثالث ، أو توزيع المفاتيح قبل بدء الإرسال .

History of Public-Key Cryptography لمحه تاريخية عن التشفير بالطريقة غير متناظرة

في منتصف عام 1970 قام البروفسور Martin Hellman والطالب الخريج Whitfield Diffie من جامعه ستانفورد بالبحث في مواضيع التشفير وخاصة مشكله إرسال المفاتيح Key Distribution Problem ، وقاموا بالتوصل إلى نتيجة هي أنه يمكن تبادل مفاتيح سريه عن طريق إرسال معلومات عامه ، حيث يمكن إرسال رسائل في خطوط غير آمنه ومليئة بالمخترقين ومع ذلك تصل أمنه ، وفي عام 1976 نشروا الورقة العلمية التي تصف الطريق التي توصلوا لها وكانت بعنوان **New Direction in Cryptography** ، وقاموا بتسميه طريق الإرسال هذه باسم Diffe-Hellman واختصارا **DH** .

وفي عام 1977 قام البروفيسور Ron Rivest من معهد MIT ، مع زملائه , Adi Shamir Len Adleman بالاهتمام بهذه الخوارزمية DH ، وقاموا بتطبيقها لتكون أول خوارزمية من نوع Public key Cryptography وتمت تسميتها باسم **RSA** (الحرف الأول من كل اسم) .

وفي عام 1985 ، قام Neal Koblitz من جامعه واشنطن و Victor Miller من مركز بحوث تابع لـ IBM ، بالبحث في فرع من فروع الرياضيات غير منتشر في ذلك الوقت وهو **elliptic curves** وذكروا أنه يمكن الاستفادة منه في تطبيق تشفير من نوع public key Cryptography ، ومع بدايه التسعينات بدأ هذا النوع من الخوارزميات بالظهور .

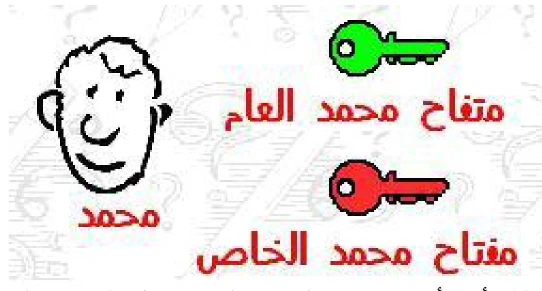
وبعد ذلك بدأت العديد من الخوارزميات بالظهور وبالانتشار ، لكن أكثر الخوارزميات انتشارا :

RSA
ECDH - Elliptic Curve Diffie-Hellman
Algamal

التوقيع الرقمي وخلاصات الرسائل *The Digital Signature and Message Digests*

نأخذ مثال بسيط لكي يوضح طريقه التوقيع الرقمي ، **ونترك التفاصيل الأخرى إلى صدور النسخة النهائية .**

صديقنا محمد لديه مفتاحين ، احدهما مفتاح عام والآخر مفتاح خاص (التشفير بالمفتاح غير المتناظر)



الآن هو يريد إرسال رسالة لأحد أصدقائه ، كل ما عليه إرسال المفتاح العام (يكون موجه للجميع) ، أما المفتاح الخاص فيحتفظ به لنفسه ، المفتاح العام يستخدم للتشفير والخاص لفك التشفير .



في حاله أراد محمد إرسال رسالة إلى علي وكان يريد أن استخدام التوقيع الرقمي (**الذي عن طريقه يستطيع علي التأكد من أن محمد هو مرسل الرسالة ، أيضا يمكن معرفه أي تغيير حصل على الرسالة أثناء إرساله**) ، كل ما على محمد هو أن يأخذ الرسالة بعد كتابتها ويدخلها إلى احد الدوال الهاشية (مثل md5 و SHA-1 وغيرها) لكي يخرج الناتج (يسمى بالهاش أو message digest).



بعد ذلك يقوم محمد بأخذ هذا الهاش ويقوم بتشفيره باستخدام المفتاح الخاص به ، وهكذا يحصل محمد على التوقيع



الآن يقوم محمد باضافه التوقيع إلى الرسالة التي يريد إرسالها .. ويرسلها إلى الأخ علي



وصلت الرسالة إلى علي ، يقوم علي (أو البرنامج الذي يستخدمه) بفك تشفير التوقيع (الناتج هو الهاش) باستخدام المفتاح العام لمحمد ، في حال انفك بشكل صحيح ، يكون علي قد عرف أن المرسل هو محمد وليس أي احد آخر..

أيضا يقوم بتطبيق الدالة الهاشية (التي طبقها محمد) على الرسالة ، في حال تساوت مع الهاش ، اذا يكون علي قد عرف أن الرسالة لم تتغير أثناء إرسالها..

الخاتمة

وهكذا نكون قد وصلنا إلى نهاية هذا الكتيب البسيط ، عسى أن تكونوا أن استفدتوا منه سواء حاليا أم مستقبلا ، واعذروني على التقصير ، ولا تنسوني من دعوه صالحه .

ونسأل الله تبارك وتعالى ، أن يكون هذا العمل خالصا لوجهه الكريم وأن يعيننا على تعلمه وتبليغه بإذنه ، إنه ولي ذلك والقادر عليه.

وأخيرا ، إن كان من صواب فمن الله تعالى ، وإن كان من خطأ فمن أنفسنا والشيطان.

وصلي اللهم وسلم وبارك على نبينا محمد وعلى اله وصحبه أجمعين.

وآخر دعوانا أن الحمد لله رب العالمين.
والسلام عليكم ورحمه الله وبركاته.

قائمه المصادر والمراجع :

The Laws of Cryptography with java code , by Neal R.Wanger

Introduction to cryptography with Java applets, David Bishop

RSA Security's Official Guide to Cryptography , Steve Burnett and Stephen Paine

www.wikipedia.com

وجدي عصام عبد الرحيم
2 – 11 – 2007
SudanGeek@hotmail.com
WajdyEssam@hotmail.com